ALFAMA CURSOS

alfamacursos.com.br

# Alfama Cursos

**Antônio Garcez**
**Fábio Garcez**
Diretores Geral

**Antônio Álvaro de Carvalho**
Diretor Acadêmico

**MATERIAL DIDÁTICO**

**Produção Técnica e Acadêmica**

**Marcela Menezes Flores**
Coordenadora Geral

**Patrícia Queiroz de Meneses**
Coordenadora Pedagógica

**José Alves Correia Neto**
**Renata Jacomo Viana**
Autoria

**Gabriella Caroline Teles Silva**
**Sabina Regina Conceição Santos**
Revisão Textual

**Rafael Rezende de Farias**
Editoração

Todos os direitos reservados e protegidos pela Lei 9.610 de 19/02/98.
É proibida a reprodução total ou parcial, por quaisquer meios, sem autorização prévia, por escrito, da ALFAMA CURSOS.

**alfamacursos.com.br**

# Fundamentos de Engenharia de Software

# Apresentação do Curso

Olá caro aluno!

Seja bem-vindo ao curso de Fundamentos de Engenharia de Software, sabendo que é uma disciplina de extrema importância para o seu conhecimento. Esta apostila vem complementar as videoaulas.

Lembre-se que apenas pela prática será possível ser um bom programador.

# Apresentação do Professor

**FÁBIO GOMES ROCHA** é formado em Sistemas da Informação e pós-graduado em Engenharia de Sistemas. Atualmente é membro da ABNT (Associação Brasileira de Normas Técnicas), fazendo parte do Comitê 21 - Engenharia de Software e do Comitê 27 - Segurança da Informação. Leciona Técnicas de Desenvolvimento de Sistemas e Linguagem de Programação e na Educação a Distância já trabalhou como desenvolvedor, professor autor e tutor em cursos universitários e técnicos, além de possuir publicações nesta área desde 2006. Além disso, atua na área de Tecnologia da Informação e Comunicação com Desenvolvimento de Sistemas há 17 anos.

Em 2004 publicou um livro sobre Tecnologia para Gestão, intitulado "Secretariado: do Escriba ao Web Writer".

# Componente Curricular

**EMENTA:**

Introdução a Sistemas; Processos de Engenharia de Software; Modelo Cascata; Prototipação; Modelo Espiral; Teoria Sistêmica; Engenharia de Requisitos; Elicitação de Requisitos; Técnicas de Elicitação de Requisitos; Análise de Sistemas; Projeto de Sistemas; Controle x Agilidade; Software de Engenharia de Requisitos; Software UML; Software para Projetos; Software para Prototipação; Software para Gestão Ágil; Metodologia Ágil na Prática; Aplicando Metodologia Ágil; Ferramentas RAD; Utilizando UML – Caso de Uso; Utilizando UML – Classe; Utilizando UML – Diagramas; Outras Ferramentas.

**PÚBLICO-ALVO:**

Alunos oriundos do Ensino Médio e universitários que queiram complementar sua formação profissional.

**OBJETIVOS GERAIS:**

- Aprofundar os estudos sobre Engenharia de Software.
- Discorrer sobre a natureza dos sistemas.
- Reconhecer metodologias de desenvolvimento de sistemas.

**HABILIDADES E ATITUDES ESPERADAS:**

- Descrever os paradigmas de Engenharia de Software.
- Identificar ferramentas Case.
- Reconhecer as ferramentas de Engenharia de Software.
- Demonstrar conhecimento sobre análise de sistemas.

# Índice


Capítulo 1 - Introdução a Sistemas ..... 9
1.1 - Exercício Proposto ..... 11
Capítulo 2 - Processo de Engenharia de Software ..... 12
2.1 – Sobre os Processos de Engenharia de Software ..... 12
2.2 - Exercício Proposto ..... 14
Capítulo 3 - Modelo Cascata ..... 15
3.1 - Levantamento de Requisitos e Análise ..... 15
3.1.1 - Projeto de Sistema ..... 16
3.1.2 - Implementação ..... 16
3.1.3 – Testes ..... 16
3.1.4 – Implantação ..... 17
3.1.5 – Manutenção ..... 17
3.2 - Exercício Proposto ..... 18
Capítulo 4 – Prototipação ..... 19
4.1 - Exercício Proposto ..... 20
Capítulo 5 – Espiral ..... 21
Capítulo 6 - Teoria Sistêmica ..... 22
6.1 - Exercício Proposto ..... 24
Capítulo 7 - Engenharia de Requisitos ..... 25
7.1 - Processos de Engenharia de Requisitos ..... 25
7.2 - Exercício Proposto ..... 27
Capítulo 8 - Elicitação de Requisitos ..... 28
8.1 - Exercício Proposto ..... 29
Capítulo 9 - Técnicas de Elicitação de Requisitos ..... 30
9.1 – Entrevista ..... 30
9.2 – Brainstorming ..... 30
9.3 - Uso de UML ..... 30
9.4 - Exercício Proposto ..... 31
Capítulo 10 - Análise de Sistemas ..... 32
10.1 - Exercício Proposto ..... 33
Capítulo 11 - Projeto de Sistemas ..... 34
11.1 - Exercício Proposto ..... 36
Capítulo 12 - Controle X Agilidade ..... 37
12.1 - Exercício Proposto ..... 38
Capítulo 13 - Software para Engenharia de Requisitos ..... 39
13.1 - Exercício Proposto ..... 41
Capítulo 14 - Software para UML ..... 42
14.1 - Exercício Proposto ..... 44
Capítulo 15 - Software para Projetos ..... 45
15.1 - Exercício Proposto ..... 47
Capítulo 16 - Software para Prototipação ..... 48
16.1 - Exercício Proposto ..... 50
Capítulo 17 - Software Para Gestão Ágil ..... 51
17.1 - Exercício Proposto ..... 52
Capítulo 18 - Metodologia Ágil na Prática ..... 53
18.1 - Exercício Proposto ..... 55
Capítulo 19 - Aplicando a Metodologia Ágil ..... 56
19.1 - Exercício Proposto ..... 58
Capítulo 20 - Ferramentas RAD ..... 59
20.1 - Exercício Proposto ..... 61
Capítulo 21 - Utilizando Caso de Uso ..... 62
21.1 - Exercício Proposto ..... 64
Capítulo 22 - Utilizando UML – Classe ..... 65
Capítulo 23 - Utilizando UML – Diagramas ..... 67
23.1 - Exercício Proposto ..... 69

Capítulo 24 - Outras Ferramentas ..... 70
Respostas dos Exercícios Propostos ..... 72
Referências ..... 75

## 1 - INTRODUÇÃO A SISTEMAS

A evolução tecnológica nas últimas décadas trouxe a necessidade de melhor planejamento, qualidade e agilidade no processo de desenvolvimento. A partir da década de 80, os microcomputadores iniciaram uma revolução, chegando a diversos mercados. O Hardware passou a custar menos, a ter poucos ou quase nenhum erro e o Software começou a ser a ferramenta crucial para as empresas. Porém, a complexidade dos programas começou a aumentar, o tempo de desenvolvimento também, e as falhas começaram a ser um problema, tornando o custo do sistema cada vez maior.

A rápida evolução tecnológica e a redução de custos de Hardware trouxeram novos desafios: como reduzir os custos de desenvolvimento de sistemas e concluir o desenvolvimento de forma mais rápida.

Os sistemas desenvolvidos sem planejamento e sem controle, não chegam ao mercado, ou quando chegam trazem maiores problemas do que solução. A exemplo disso temos o incidente com o Therac-25, um sistema de Raios-X, que entre 1985 e 1987 tiveram problemas por ministrar doses elevadas de radiação, a pelo menos seis pacientes, matando alguns e incapacitando outros. Segundo a investigação feita pela Food and Drug Administration (FDA), os problemas do Therac eram entre outros: documentação pobre, falta de especificação e de plano formal de teste, além de ser considerado um projeto amador de Software para controle de dispositivo, com a falta de sincronização e limitações no compartilhamento de dados.

A falta de planejamento cria problemas no projeto de Software e, que muitas vezes torna-o inviável. Parte do problema está ligada à necessidade de agilidade e ao fato de que o desenvolvimento de Software é uma atividade relativamente nova. O desenvolvimento com lógica binária mostra-se uma prática recente, saída de meados da década de 60, quando o programador responsável pelo desenvolvimento de um produto atuava sem tempo pré-definido, visto que, na época, a produção era para âmbito acadêmico. A globalização começou a exigir da indústria práticas diferentes e a conexão com o mundo através da Internet encurtou prazos e exigiu mais dedicação na implementação, em que a segurança deveria ser primordial (devido ao crescimento da World Wide Web), sem abandonar uma boa qualidade de desenvolvimento e agilidade. Esta agilidade trouxe diversos problemas: segundo o Standish Group, 13,1% dos projetos possuem fatores críticos de requisitos incompletos; segundo estudos da IBM, 55% dos sistemas custam mais do que o esperado, 68% ultrapassam o cronograma original e 8% devem ser reprojetados; a estatística do Bureal Of Labor Statistics concluiu que a cada seis novos sistemas colocados em operação, dois são cancelados, e ainda 75% dos sistemas no seu todo representam falhas operacionais. Sabendo-se desses percentuais de insucesso dos projetos, devido à falta de planejamento, a indústria sentiu a necessidade da criação de processos adequados para gerenciar as atividades dos mesmos, onde não só o desenvolvimento deveria ser analisado, mas seu âmbito de pré e pós-produção. Portanto, a necessidade de métricas fez a indústria criar modelos de padronização de produção.

A experiência inicial na construção de sistemas foi extremamente alterada. Atualmente, o estudo da criação de um Software é uma das etapas que mais demanda tempo, mesmo que tenha como objetivo a melhoria de processo e redução da curva de aprendizado da empresa. A forma que antes era utilizada foi insuficiente para acompanhar o desenvolvimento da indústria, que busca um tempo determinado e condições bem estabelecidas de qualidade, garantindo a confiança dos clientes para com os seus produtos.

Desta forma, é importante que se entenda que um dos primeiros passos para o desenvolvimento qualitativo de Software é entender o que se quer produzir, e neste sentido a Engenharia de Software auxilia o processo de planejamento e desenvolvimento

do Software.

A Engenharia de Software possibilitou a criação em menor tempo de novos aplicativos mais complexos e com maior qualidade. O século XXI mostrou, com a melhoria das aplicações web e a evolução do conceito da computação na nuvem (cloud computing), onde tudo que é aplicativo ou serviço está disponível de forma remota, em qualquer lugar, o quão rápido o ser humano pode chegar quando os processos são organizados, por mais básicos que sejam. Portanto, esta apostila busca tratar o conhecimento acerca do desenvolvimento de Software: suas partes, a integração entre elas e a geração de um produto final com qualidade.

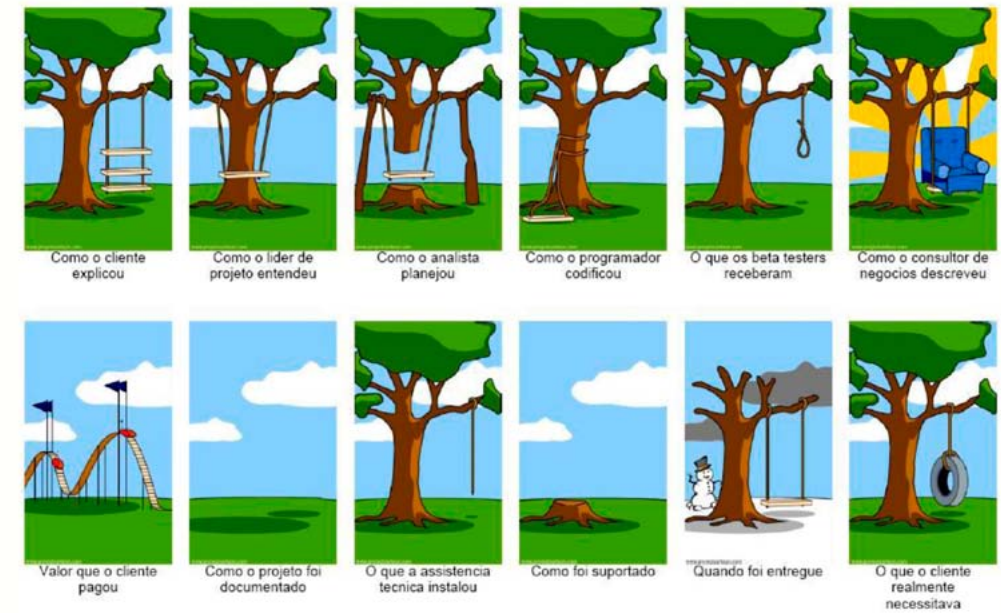

Como o cliente explicou
Como o líder de projeto entendeu
Como o analista planejou
Como o programador codificou
O que os beta testers receberam
Como o consultor de negocios descreveu
Valor que o cliente pagou
Como o projeto foi documentado
O que a assistencia tecnica instalou
Como foi suportado
Quando foi entregue
O que o cliente realmente necessitava

Figura 1: Processo de Produção de Software.

## Fica a dica!

Planejar é muito mais do que apenas descrever o que o cliente quer, então a dica é fazer o cliente participar do desenvolvimento, assim evita o problema apresentado na Figura 1.

## Recordando

A Engenharia de Software surgiu para ajudar na solução de problemas, como os ocorridos com o Therac-25 e outros. Esta foi necessária, graças ao surgimento do PC em 1980 que demandou por novos Softwares crescendo ainda mais os problemas de desenvolvimento.

### 1.1 - EXERCÍCIO PROPOSTO

1) Qual o principal problema em Projetos de Software?

_____
_____
_____

## 2 - PROCESSO DE ENGENHARIA DE SOFTWARE

A Engenharia de Software (ES) é um ramo da computação que tem como principal objetivo solucionar os problemas práticos do mundo real através da aplicação de conhecimentos científicos. O termo Engenharia de Software surgiu em 1968 através de eventos computacionais em várias atividades da sociedade. Entre os vários problemas conhecidos, há o estouro de prazo e orçamento, pois grandes projetos representavam grandes dificuldades de conclusão. Ainda que pudessem concluir um projeto, havia um grave problema de manutenção devido à documentação rara, o que criava um caos para o processo de melhoria do Software. A Engenharia de Software tornou-se uma peça-chave para a construção de sistemas, pois os projetos se tornaram cada vez mais complexos e a alta demanda do mercado por tecnologia de informação e comunicação obrigou a adoção de novas práticas de desenvolvimento. A Engenharia de Software faz uso, em geral, da teoria sistêmica, é um dos componentes mais importantes no estudo da análise e projeto de Software.

Processo é um conjunto de atividades relacionadas e executadas passo a passo ou em paralelo. Todo processo tem início, meio e fim e possui saídas bem definidas. O que difere um processo de um projeto é o prazo de conclusão. Em um projeto o prazo está determinado, no processo é algo contínuo, mas sempre tendo início, meio e fim. Os processos de Engenharia de Software envolvem tarefas de projeto, planejamento, implementação, implantação, teste, manutenção, documentação e etc. O processo é de certa forma uma maneira de se realizar operações seguindo normas. Existem métodos para melhorias de processos, como vimos na parte de SGQ (Sistema de Gestão da Qualidade) sobre o PDCA. Eles visam tornar o processo eficaz, produzindo resultados visados, eficientes, buscando consumir poucos recursos e resultados efetivos e vantajosos em relação aos gastos dos processos de Engenharia de Software.

A Engenharia de Software é um método pelo qual se realiza determinadas tarefas seguindo padrões, ou seja, o desenvolvimento baseia-se em ações sistemáticas e não em ações improvisadas e sem nexo.

### 2.1 – SOBRE OS PROCESSOS DE ENGENHARIA DE SOFTWARE

Devido às novas tecnologias de Hardware e Software, um projeto torna-se obsoleto antes mesmo de ser concluído. O Software é um produto cujo desenvolvimento depende muito do desenvolvedor humano passível de erros, não é um processo automatizado, como em linhas de produção de carros, por exemplo. Por este motivo, processos de desenvolvimento ajudam a criar Softwares cada vez mais eficientes, em diversas fases. Processo de Software ou Processo de Desenvolvimento de Software é constituído por um conjunto de atividades, com o objetivo de produzir um produto: o Software. O Processo de Desenvolvimento de Software tem se tornado cada vez mais complexo devido às novas necessidades de recursos e soluções que os sistemas devem atender. Os sistemas modernos baseiam-se em uma grande quantidade de linhas de código. Por se tratar de um processo altamente manual, ou seja, não automatizado, dependente da capacidade humana para o seu desenvolvimento, depende da criatividade e do comprometimento para que o produto final possua qualidade. Buscando evitar a criação de produtos sem qualidade, é possível utilizar um modelo de desenvolvimento de Software que se adeque à empresa, e desta forma atender melhor as necessidades dos clientes internos/externos. Os modelos de desenvolvimento são conhecidos como Ciclos de Vida de Desenvolvimento de Sistemas ou SDLC (da sigla em inglês para Systems Development Life Cycle) abordados nos próximos capítulos.

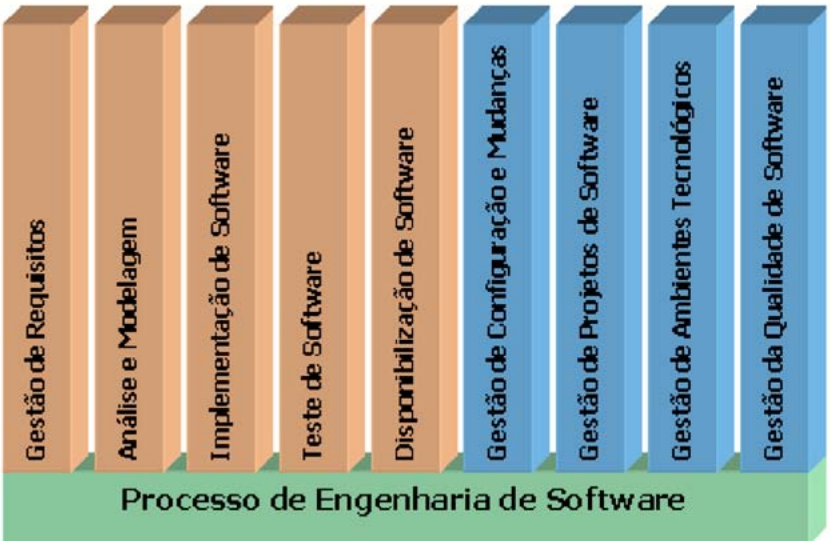



Figura 2: Processos de Engenharia de Software.

## Recordando

O Processo de Engenharia de Software surgiu da necessidade de melhorar os produtos de Software criados devido ao excesso de falhas e problemas no projeto, sendo o processo um conjunto de fases dependentes ou independentes que tem objetivos em comum: a produção de um produto.

### 2.2 - EXERCÍCIO PROPOSTO

1) O que é Engenharia de Software?

_____

_____

_____

## 3 - MODELO CASCATA

O modelo em cascata é classificado como Ciclo de Vida de Desenvolvimento de Sistemas ou SDLC (da sigla em inglês para Systems Development Life Cycle). Trata-se de uma abordagem clássica que possui uma sequência lógica de passos no desenvolvimento de qualquer sistema. Este modelo divide-se nas fases: Levantamento de Requisitos e Análise, Projeto de Sistema, Implementação, Testes, Implantação de Sistema e Manutenção.

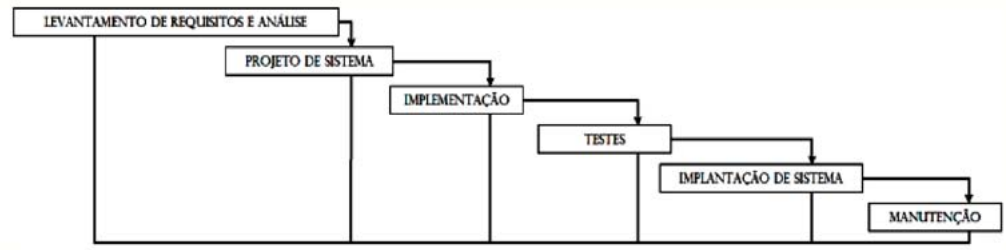



Figura 3: Ciclo de Vida Cascata.

Todas as etapas do modelo em cascata são interligadas, porém possuem uma finalidade específica. Esta abordagem é rigorosa, ou seja, há delimitação clara de início e fim de fase, não havendo iterações. Com exceção da fase de concepção, cada etapa inicia-se quando a anterior é finalizada. Após a definição de requisitos, as modificações deverão ser evitadas. Qualquer erro pode ser fatal para um projeto, pois da mesma forma que uma etapa bem-sucedida influencia no desempenho de outra, um erro cometido propaga-se em todos os níveis do modelo em cascata. Veremos nos tópicos a seguir, os detalhes de cada etapa do modelo em cascata.

### 3.1 - LEVANTAMENTO DE REQUISITOS E ANÁLISE

Esta é a fase do Ciclo de Vida de Desenvolvimento de Software que determina os requisitos de Software através da identificação do problema apresentado pelo cliente. O analista deverá cumprir seu papel, ou seja, solucionar os problemas através do uso de técnicas e ferramentas para:

I - Entrevistar o cliente.
II - Entender o problema.
III - Documentar o que foi entendido.
IV - Validar a documentação produzida com o cliente.

A entrevista é umas das melhores maneiras de se obter informação sobre o problema a ser resolvido. Esta etapa é considerada crítica porque normalmente o entrevistado é o cliente, um dos stakeholders mais importantes do projeto. Normalmente, este indivíduo pouco sabe expressar suas necessidades, enquanto o analista pouco conhece as técnicas para extrair informações relevantes para o projeto. As questões levantadas deverão extrair exclusivamente conhecimento sobre os requisitos. Comentar precocemente sobre tecnologia é desnecessário e muitas vezes nem é recomendável durante uma entrevista. O analista deverá ter acima de tudo, simplicidade ao conhecer o nível do cliente em relação à sua linguagem e suas limitações. Não haverá tempo para pedir explicações sobre tudo que acontece na empresa, portanto deve-se enfatizar o processo para solucionar o problema atual. Para melhorar a comunicação com os stakeholders, o analista pode usufruir de diagramas de caso de uso ou até mesmo mindmaps (mapas mentais). Entender o problema é essencial, portanto esta iniciativa é considerada o núcleo da análise.

Uma técnica adotada em grandes centros de desenvolvimento de Software para entender o problema são as sessões de brainstorming. Trata-se de um grupo de discussão que possui como integrantes todas as partes envolvidas no projeto (também stakeholders). Estes são desenvolvedores, gerentes de projetos, arquitetos de Software, analistas, entre outros. Muitos consideram a primeira reunião com o cliente como um brainstorming, porém este

termo está mais associado a uma metodologia utilizada por equipes de desenvolvimento para entender as informações extraídas durante a primeira reunião com o cliente. As reuniões ocorrem do início ao fim do projeto e envolvem diferentes perspectivas e abordagens livres.

### 3.1.1 - PROJETO DE SISTEMA

A fase de projeto ou arquitetura de um sistema envolve o mapeamento de variáveis essenciais para o desenvolvimento de Software. A principal diferença para a fase de análise é a definição da tecnologia, agora considerada indispensável. Aspectos como ferramentas, plataformas e linguagens de programação são considerados por uma equipe de arquitetos de Software. Considera-se então, Dados, Processos, Hardware ou Software e Informação. A definição de requisitos ocorrida durante a análise é base para o projeto que contempla vários níveis de Hardware e Software. Por exemplo, um cliente deseja ter um sistema web rápido, porém altamente seguro. A partir dessa necessidade, toda a equipe deverá definir como o sistema será desenvolvido e quais são as condições de uso: conexão criptografada e internet banda larga, além de Hardware para que este sistema seja operado satisfatoriamente. Normalmente, quando compramos um Software de prateleira vemos na caixa do produto as recomendações de uso do fabricante (sistema operacional, processador, memória e espaço em disco). Com os requisitos definidos e devidamente documentados, todo o processo de produção de Software ocorrerá com o mínimo de erros. Por ter uma grande quantidade de profissionais envolvidos, o projeto é considerado como uma das fases mais caras e complexas do processo de produção de Software. Para diminuir custos é necessário investigar o legado de tecnologia da empresa. Ou seja, há componentes ou serviços já existentes ou dispensáveis. Determinado Software antigo pode oferecer satisfatoriamente uma determinada função que já torna uma futura implementação deste aspecto não mais necessária em um novo sistema oferecido pelos desenvolvedores. Para este tipo de caso recomenda-se oferecer um mesmo componente ou serviço apenas quando solicitado pelo cliente. Tudo deverá ser criteriosamente documentado para esquematizar todo processo de implementação.

### 3.1.2 - IMPLEMENTAÇÃO

A implementação é a tão esperada fase de desenvolvimento das linhas de código. Pessoas leigas têm uma opinião bastante equivocada ao considerar a implementação como a principal fase de desenvolvimento de Software. As equipes de desenvolvimento só podem desenvolver exatamente o que foi solicitado se as fases de análise e projeto forem bem-sucedidas. Esta etapa poderá ser uma construção a partir do zero ou depuração (correção de erros). Há critérios específicos que deverão ser observados durante a codificação como a legibilidade, reuso e flexibilidade.

A legibilidade deve-se ao código escrito que deverá ser de fácil compreensão. Artifícios como documentação de código possibilitam não só o melhor entendimento de código como o seu reuso. Obviamente a empresa especializada em um tipo de sistema particular pode aproveitar alguns componentes para outros projetos. A flexibilidade é definida pela capacidade de aceitação que um determinado cliente possui em aceitar novas funcionalidades sugeridas.

Através desta fase é perceptível a introdução de aspectos de qualidade, a partir do momento que as diretrizes de qualidade (requisitos) são seguidas pelos desenvolvedores. Há ferramentas que podem acelerar o processo de desenvolvimento. Um tipo bem comum é o CASE (Computer Aided Software Engineering ou Engenharia de Software Auxiliada por Computador) que oferece um auxílio automatizado aos desenvolvedores de sistemas através de modelos. Alguns dos componentes providos pela ferramenta CASE são a assistência à documentação e o suporte à análise e gerenciamento de projetos. Um Framework que possui estes componentes, além da capacidade de gerar código pode ser considerado uma ferramenta CASE. Entre as vantagens percebe-se visivelmente a produtividade, porém em processos maiores é difícil e complexo o controle das linhas de código geradas.

### 3.1.3 – TESTES

A fase de testes de Software é mais uma área de conhecimento que se compromete com a qualidade. Testar é a atividade de analisar e executar aplicações com a intenção de

verificar se o comportamento dos sistemas é o esperado. O objetivo dos testes de Software vai bem além de revelar defeitos. A literatura traduz um defeito (ou bug) como um passo, processo ou definição de dados incorretos. Um defeito poderá causar um erro (um estado inconsistente), que resultará em falhas (comportamentos que diferem do esperado). Diferente de uma revisão simples, ou análise estática de código, o teste compreende vários mecanismos baseados nos conceitos de Engenharia de Software. Um caso de teste é composto por dados de entrada e resultados esperados. Para um ou mais casos de testes há um oráculo, um mecanismo que decide sobre a correção de uma execução. Porém, há oráculos que são complicados devido ao desconhecimento dos resultados, ou resultados esperados difíceis de ser avaliados ou até mesmo devido à grande quantidade de dados. Para resolver este e alguns dos problemas frequentes na definição de estratégias de testes, há critérios de testes que envolvem uma seleção de um domínio e elaboração de caso de testes.

Portanto, a etapa de testes é um processo de verificar e validar o que em uma aplicação de Software atende a regras técnicas e de negócio que conduzem ao projeto em implementação.

### 3.1.4 – IMPLANTAÇÃO

A implantação de um sistema depende da homologação do produto que foi testado previamente. Serão realizados testes de instalação e integração para garantir que o produto seja operacional conforme os requisitos do cliente. A conformidade também deverá estar associada às normas técnicas internacionais e padrões do fabricante. É por esta razão que muitas empresas de Software, como a Oracle, solicitam que sejam realizadas inspeções para certificar que seu cliente siga as recomendações de utilização e segurança. Há também um aspecto importante, que é definir qual estratégia de implantação.

Esta estratégia deverá considerar a infraestrutura básica, por exemplo, a existência de uma máquina virtual ou de um ambiente de execução dos serviços da aplicação homologada. A existência de máquinas (servidores) de produção e homologação faz parte de uma estratégia de implantação de um produto de Software em uma empresa. Além da instalação, também se deve pensar em treinamento para capacitar o cliente e garantir uma boa aceitação do produto.

### 3.1.5 – MANUTENÇÃO

A manutenção consiste na correção de todo tipo de componente de Software produzido (desde conjuntos de linhas de código até interfaces gráficas e documentações). Neste momento é comum encontrar vários erros cometidos por equipes nas fases de desenvolvimento anteriores. Por exemplo, uma aplicação antiga com documentação pobre proveniente da fase de análise confunde toda uma equipe encarregada na manutenção do sistema, depreciando a qualidade deste. A legibilidade do código da implementação também é um fator muito comum causador de problemas de manutenção. Portanto, a manutenção só funcionará satisfatoriamente se as fases precedentes forem conduzidas apropriadamente.

O propósito da manutenção está dividido em correção, adaptação, aperfeiçoamento e prevenção. A correção ocorre quando um programador corrige erros de código (bugs), enquanto a adaptação habilita uma aplicação a novas plataformas (uma migração do ambiente web para o mobile). O aperfeiçoamento lida com a mudança de requisitos do usuário causando eventuais alterações na aplicação (melhorias na interface gráfica para aumentar o entendimento do usuário). E por último a prevenção, um programador pode melhorar a legibilidade de seu código para aumentar a capacidade de manutenção do sistema.

Há aspectos do sistema que são importantes e devem ser considerados antes de começar uma manutenção. Sistemas bem antigos dificultam ou até mesmo inviabilizam uma manutenção. Aplicações financeiras programadas em COBOL são constantemente mantidas, pois sai mais barato manter o ambiente em que estas aplicações são operadas do que desenvolver uma nova com os mesmos serviços. Mas quando a opção é realizar uma migração de tecnologia, a equipe vale de metodologias para entender como funciona o sistema alvo, para depois oferecer a manutenção. Uma técnica muito comum é a Engenharia Reversa, uma desconstrução completa do sistema para gerar modelos descritivos de desenvolvimento.

## 3.2 - EXERCÍCIO PROPOSTO

1) Quais são as fases do Ciclo de Vida Cascata?

_____
_____
_____

## 4 – PROTOTIPAÇÃO

Apesar do modelo em cascata ser bastante claro em relação às fases de desenvolvimento distintas, sua desvantagem é a grande suscetibilidade a erros. Outra desvantagem, até que o sistema seja testado pelo usuário final, não há noção alguma se o produto está de acordo com os requisitos.

O tempo para chegar até chegar à fase de testes do modelo clássico (em cascata) é longo. Pensando neste problema, um processo foi elaborado para representar de forma mais agradável o produto em construção. Este processo é denominado prototipação. Ele foi criado exatamente para confrontar o ciclo de vida clássico oferecendo uma melhor forma de comunicação entre usuários e desenvolvedores. O principal objetivo da prototipação é reduzir as falhas de requisitos. Uma análise de requisitos será realizada, que conduzirá a um projeto rápido e, então, a construção de um protótipo. O estágio final do processo é a avaliação do protótipo. Este protótipo será submetido a um tunning (ajuste fino) que envolve modificações de interfaces ou inclusão de requisitos para atender às necessidades do usuário.

Há tipos de prototipações que podem ser destacados, como a evolucionária e a exploratória (ou descartável). A prototipação evolucionária tem o propósito de evoluir o sistema gradualmente até chegar a um sistema funcionando. E a prototipação exploratória cria vários protótipos de descarte para validar ou derivar requisitos. Tudo parece funcionar bem, mas a construção de protótipos de sistemas computacionais é tão custosa quanto dar manutenção a milhões de linhas de código. Em sistemas complexos não é recomendado criar protótipos visuais que não podem ser aproveitados por equipes de desenvolvimento para a construção de interfaces gráficas. Há ocasiões que necessitam de um funcionário específico para elaborar as interfaces gráficas. Simplesmente porque um desenvolvedor comum pode ser superficial nos detalhes de usabilidade. É por esta razão que muitas equipes de desenvolvimento criaram estratégias de programação, geração e aproveitamento de código. A figura 4 apresenta as fases da prototipação.

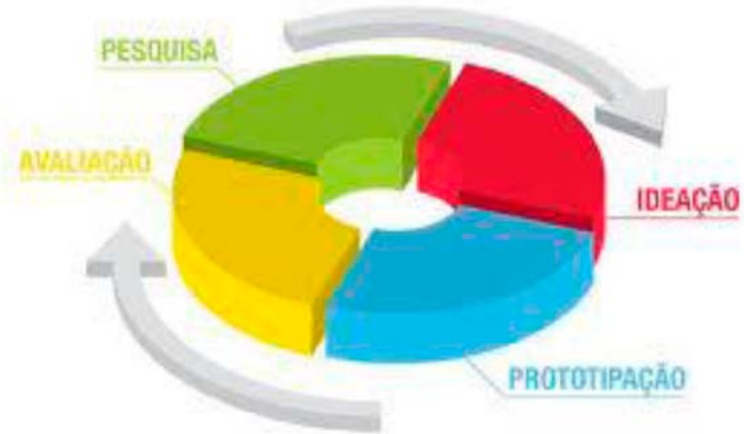



Figura 4: Prototipação.

## 4.1 - EXERCÍCIO PROPOSTO

1) Qual a principal vantagem da Prototipação?

_____
_____
_____

## 5 – ESPIRAL

O modelo em espiral também é classificado como um Ciclo de Vida de Desenvolvimento de Sistemas ou SDLC. É uma combinação da sistemática do Ciclo de Vida em Cascata com a Prototipação. Cada iteração da espiral é divida nas seções de Análise, Projeto, Implementação e Testes. A análise é similar a de outros paradigmas, é a configuração de um objetivo. O projeto possui os mesmos objetivos abordados anteriormente no Ciclo de Vida Clássico, e configura-se como um processo de avaliação e redução de riscos. E finalmente, os testes servem como validações do processo produtivo e como planejamento de próximas fases evolutivas do desenvolvimento de Software.

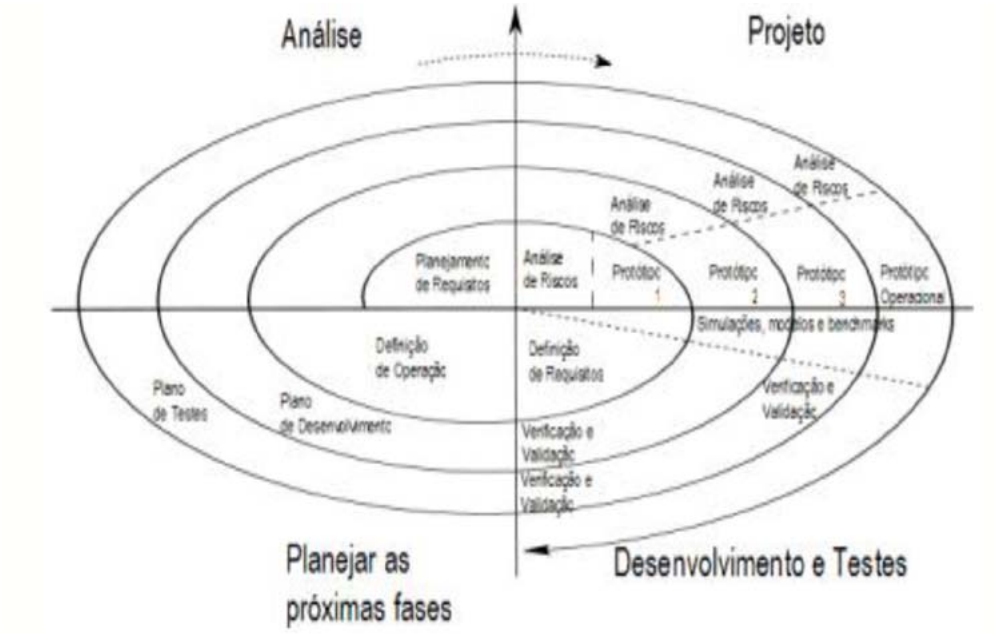



Figura 5: Ciclo de Vida Espiral.

A proposta do modelo em espiral é o desenvolvimento evolucionário, no qual não é necessário definir previamente todos os requisitos do sistema. Apenas os requisitos de prioridades devem ser inicialmente definidos e implementados. Ao final do primeiro ciclo, as equipes de desenvolvimento terão seções ainda menores para implementar.

## 6 - TEORIA SISTÊMICA

Embora a origem da teoria de sistemas seja, por alguns estudiosos do tema, atribuída à época sumeriana há mais de 2.500 a.C (Lieber, s/d), ela surge, efetivamente, com o trabalho do biólogo austríaco Ludwig Von Bertalanffy, em seus trabalhos publicados em 1950 e 1968. Os sistemas podem ser definidos como o conjunto de elementos interrelacionados com um objetivo em comum.

Segundo os teóricos atuais sobre o tema (Araújo, V.R.H., Katz D. e Kahn R. L), um sistema deve possuir as seguintes características:

- Importação de energia.
- Transformação.
- Produto.
- Sistemas como ciclos de eventos.
- Entropia negativa.
- Insumo de informação, realimentação negativa e processo de codificação.
- Estado estável e homeostase dinâmica.
- Diferenciação.
- Finalidade.

Além dessas características, os sistemas seguem as normas e leis, independente de área:

- Todo sistema é composto de subsistemas.
- Todo sistema é parte de um sistema maior.
- Quanto maior a divisão de subsistemas, maior a necessidade de coordenação de partes.
- Homeostase.
- Sinergia.

Uma abordagem sistêmica é uma maneira de resolver problemas sob o ponto de vista da Teoria Geral de Sistemas, facilitando o entendimento do problema e criando soluções mais eficazes. Muitas soluções surgem quando observamos um problema como um sistema e, desta forma, são constituídos por elementos, com relações, objetivos e um meio ambiente. Pode-se notar que sob a nova concepção, apesar de dividir os problemas em partes para simplificar a solução é importante o entendimento do todo.
Para abordar uma situação como sistema pode-se adotar os seguintes princípios:

- Dividir para conquistar, ou seja, dividir problemas maiores em problemas menores, ou o que chamamos de atomismo.
- Identificar as partes do sistema, lembrando que a soma das partes geram o todo, mas o todo é maior que a soma das partes.
- Atentar-se para os detalhes.
- Olhar para o todo, ou seja, o macroprocesso é importante para entender como os microprocessos se relacionam.
- Uso de analogias.

> Dividir o todo em partes torna mais fácil a compreensão das partes menores, porém o que acontece é que se ao dividir o processo em partes menores, não se entende o todo, há uma perda inevitável de aspectos, propriedades do todo, por ocasião da redução. (ABBAGNANO, 1995)

> O sistemismo representou uma profunda revolução na história do pensamento científico ocidental. A crença segundo a qual em todo sistema complexo o comportamento de todo pode ser entendido inteiramente a partir

> das propriedades de suas partes é fundamental no paradigma cartesiano. (CAPRA, 1999, p. 41)
>
> O pensamento sistêmico (2002, p. 15), é contextual, ou seja, o oposto do pensamento analítico requer que para se entender alguma coisa é necessário entendê-la, como tal, e em um determinado contexto maior, ou seja, como componente de um sistema maior, que é o seu também chamado ambiente. Desta forma pode-se dizer que o todo é maior do que a soma das partes. (GUNTTER, 2002, p. 15)
>
> (..) "...apreender a ideia do todo e, partindo dela, considerar na faculdade pura da razão todas aquelas partes na sua relação recíproca, derivando-as do conceito do todo. Esse exame e essa garantia são possíveis apenas por meio do conhecimento mais íntimo do sistema; e os que esmoreceram na primeira investigação, que pensaram, por conseguinte, que não valia a pena adquirir esse conhecimento, não chegarão ao segundo grau, isto é, a visão do conjunto, que é um regresso sintético ao que antes foi analiticamente; não é de admirar que encontrem inconsequência em toda parte, apesar de as lacunas que eles supõem não estarem no próprio sistema, mas unicamente na marcha incoerente do pensamento deles...." (KANT)

Logo, podemos notar que o entendimento do macroprocesso é de suma importância, principalmente, no que diz respeito ao processo de desenvolvimento de sistemas com qualidade. Um sistema que não tenha interação correta entre os módulos poderá causar perda de informação, ou ainda, o sistema pode não atender às necessidades do cliente devido a um mau levantamento de requisitos, o que implica em baixa qualidade.

Na prática: Primeiro entenda o que se quer produzir, para depois dividir e entender o problema em partes; a próxima etapa é entender como as partes se comunicam, ou seja, a sinergia do sistema. O próximo passo é a homeostase, ou seja, as generalidades, descobrir o que é genérico no sistema e que pode ser reutilizado, é de suma importância, pois pode ser testado uma única vez e estará pronto para a utilização. Estes passos são executados no macroprocesso e reexecutados em cada módulo e assim por diante, até o menor passo do sistema. Desta forma, a possibilidade de conseguir atingir os objetivos iniciais do projeto é maior.

A teoria sistêmica é o caminho para atingir os objetivos de desenvolvimento.

Uma visão sistêmica auxilia o entendimento e o levantamento dos requisitos como veremos a seguir.

## 6.1 - EXERCÍCIO PROPOSTO

1) Quais são as leis que regem o Sistemismo?

_____
_____
_____

## 7 - ENGENHARIA DE REQUISITOS

A Engenharia de Requisitos é parte da Engenharia de Software e pode ser definida como o processo que visa fornecer informações de forma adequada sobre as necessidades do cliente e do negócio, organizando a criação de uma solução de acordo com o cenário indicado.

Segundo Cristel e Kang (2003) "a Engenharia de Requisitos é a etapa que deve descrever os requisitos de forma clara e concisa, além de gerenciar as mudanças durante o processo de desenvolvimento e o ciclo de vida do sistema".

Sob a definição de diversos teóricos (Cristel e Kang (2003), Blaschek (2002), Paula (2002), Peters e Pedrycz (2001), Pressman (2001), Quedas (2003)) "os requisitos visam estabelecer o conjunto de características que o Software deve ter para a aceitação por parte do cliente".

### 7.1 - PROCESSOS DE ENGENHARIA DE REQUISITOS

O processo pode ser dividido em cinco etapas conforme:

- Elicitação de requisitos
- Análise de requisitos
- Especificação de requisitos
- Validação de requisitos
- Gestão de requisitos

> O custo de mudança durante a fase de requisitos é de um a seis vezes menores do que durante a fase de desenvolvimento e de sessenta a cem vezes menores do que durante a fase de entrega do produto. Logo, pode-se dizer que a fase de Engenharia de Requisitos é crucial para o sucesso do produto. (PRESSMAN, 2001)

Pode-se solucionar os problemas através da adoção de um processo formal de Engenharia de Requisitos, que permita uma real compreensão das necessidades para o sistema, com a delimitação correta dos processos a serem implementados e a validação das necessidades do cliente.

Segundo a ISO 9001 "há uma recomendação de que os requisitos do cliente devam ser atendidos para o aumento de sua satisfação. Para tornar o processo mais simples é necessário um gerenciamento eficaz dos requisitos, viabilizando eventuais ajustes ou mudanças".

> Gerenciamento de requisitos pode ser definido como um processo que estabelece e mantém a concordância entre o cliente e os desenvolvedores durante as alterações dos requisitos do projeto, aproximando sistematicamente a elicitação, organização e documentação dos requisitos de Software. (OBERG et. al., 2001)

Os requisitos estabelecem um conjunto de características necessárias à aceitação de um Software por parte do cliente, descrevendo quais atividades o Software deve desempenhar, quais suas limitações e restrições, além de outras características não ligadas diretamente às funções desempenhadas por ele, como metas de qualidade. Pode-se, também, definir que a Engenharia de Requisitos tem como principal funcionalidade determinar o que será e o que não será feito, estabelecendo, desta forma, as restrições referentes ao sistema.

Durante o processo de gerenciamento de requisitos é necessário tomar decisões sobre os seguintes aspectos:

- **Identificação dos Requisitos:** para cada requisito é necessária uma identificação única, permitindo que possa ser efetuada a referência cruzada deste com os outros requisitos, e que ele possa ser utilizado nas avaliações de rastreamento.
- **O Processo de Gerenciamento de Mudanças:** é o conjunto de atividades que deve avaliar o impacto e o custo de uma determinada mudança.
- **Políticas de Rastreamento:** esta política define as relações entre os requisitos e o projeto de sistema, que devem ser registradas para o gerenciamento futuro.

## 7.2 - EXERCÍCIO PROPOSTO

1) Quais as etapas da Engenharia de Requisitos?

_____
_____
_____

## 8 - ELICITAÇÃO DE REQUISITOS

Para entender o que é Elicitação de Requisitos, vamos entender os dois termos separadamente.

Elicitação: significa descobrir, tornar claro e explícito.

Requisitos: segundo o Dicionário Aurélio é a condição que se deve satisfazer para alcançar um certo fim, ou ainda a exigência de ordem legal para que determinado processo possa ter andamento. O requisito é uma necessidade, uma exigência.

Logo, a Elicitação de Requisitos é a descoberta das necessidades do produto. Podemos classificar os requisitos em:

- **Requisitos do Usuário:** possui um alto nível de abstração. Os Requisitos do Usuário são as descrições dos serviços esperados do sistema, bem como suas restrições de operação.
- **Requisitos de Sistema:** processo detalhado. É a definição detalhada das estruturas de serviços e suas restrições.

Os Requisitos de Sistemas podem ser Funcionais ou Não Funcionais.

Os Requisitos Funcionais são declarações de funcionalidades que o Software deve oferecer e ainda pode estabelecer o que o Software não deve fazer em quesito de funcionalidade.

Os Requisitos Não Funcionais estão atrelados ao comportamento, como velocidade, equipamento necessário e etc.

Para a Elicitação de Requisitos são necessárias técnicas que auxiliem a descoberta das necessidades do cliente/usuário. Por ser uma fase de alta importância é necessário validar os requisitos a fim de testar se o que foi levantado é o que deve ser feito na etapa de desenvolvimento.

## 8.1 - EXERCÍCIO PROPOSTO

1) O que é Elicitação de Requisitos?

_____
_____
_____

## 9 - TÉCNICAS DE ELICITAÇÃO DE REQUISITOS

A elicitação é um dos principais fatores para o sucesso do desenvolvimento de um sistema, para tanto, é necessário que o analista conheça as técnicas que podem ser utilizadas para a Elicitação de Requisitos.

### 9.1 – ENTREVISTA

Esta é normalmente a primeira técnica a ser utilizada. É feita uma entrevista com os usuários-chaves (key users) para determinar as necessidades do sistema.

Na técnica de entrevista pode-se utilizar questionários abertos e fechados que facilitam a descoberta de informações.

Deve-se fazer a entrevista individual ou com pequenos grupos. Tem como ponto negativo o tempo e o custo, como ponto positivo tem a eficiência.

### 9.2 – BRAINSTORMING

Brainstorming é a técnica denominada tempestade de ideias, muito útil para cobrir as necessidades de um grupo de pessoas. É realizada uma reunião com um grupo de usuários, junto com o analista e o cliente. Cada um deve ter a liberdade para dar ideias sobre o sistema.

O objetivo do brainstorming é permitir que todos os usuários que vão utilizar o recurso do Software possam expressar suas ideias sobre as funcionalidades do sistema, e obter um consenso sobre o que deve ou não estar no produto.

Tem como ponto positivo a organização das ideias do grupo, identificação de conflitos de ideias e da visão de cada um sobre o produto, como ponto negativo o gerenciamento da reunião deve ser feito cuidadosamente, pois não pode perder o foco da funcionalidade.

### 9.3 - USO DE UML

Entre as técnicas utilizadas para a Elicitação de Requisitos, está a utilização de UML para especificar as necessidades. O caso de uso é um dos mais utilizados aqui, pois descreve como as pessoas interagem com os recursos do Software.

Além do caso de uso, é possível utilizar também o diagrama de atividades, que serve para exibir o fluxo de trabalho.

## 9.4 - EXERCÍCIO PROPOSTO

1) Quais são as Técnicas para Elicitação de Requisitos?

_____
_____
_____

## 10 - ANÁLISE DE SISTEMAS

A fase de análise estuda os requisitos para modelar o problema, ou seja, criar modelo de soluções do problema. Nesta fase é feita, inicialmente, a classificação das informações do levantamento de requisitos e com base nestas informações é feita a análise dos requisitos de sistemas, para assim poder criar os modelos para o sistema.

Os modelos devem ser independentes de tecnologia, para que na etapa de implementação, sejam codificados na linguagem definida, mas ao mesmo tempo em que possam ser trocados de linguagem. Nesta etapa de modelagem, cria-se a matriz de dependências de requisitos, gera os documentos necessários, entre outros.

Existem diversas técnicas de análise, destacando-se:

- **Técnicas de Análise Estruturada:** considerada atualmente obsoleta, mas foi a técnica mais utilizada do mercado de desenvolvimento. Tem como foco as funções e os eventos que afetam o sistema. Normalmente atua em três visões, sendo: funções, dados e controle. É uma técnica focada na função e não no problema.
- **Técnica Orientada a Objetos:** é mais abstrata e é focada na solução de problemas do mundo real, buscando atender melhor a demanda de análise. Os objetos tendem a se comunicar, há uma perspectiva integrada e de reaproveitamento.

## 10.1 - EXERCÍCIO PROPOSTO

1) Quais são as Técnicas de Análise mais conhecidas?

_____
_____
_____

## 11 - PROJETO DE SISTEMAS

1) "Um projeto é um conjunto temporário de esforços objetivando um resultado exclusivo".

Essa é uma redução da definição de projetos do PMBOK – um Guia Para o Gerenciamento de Projetos, editado pelo PMI – Project Management Institute, e que será apresentado na íntegra posteriormente. A intenção é desmitificar a ideia de que um projeto deva ser conduzido estritamente de acordo com esse ou aquele modelo. Os principais modelos de gerenciamento de projetos oferecem "boas práticas" que, portanto, podem coexistir no mesmo projeto. Frequentemente, as ideias apresentadas são as mesmas, tratadas sob abordagens distintas, compatíveis com cada metodologia. Caberá ao responsável pelo projeto decidir quais práticas adotar.

2) "Os projetos são temporários".

O protótipo de um produto é fruto de um projeto. Sua produção em escala é uma operação. O projeto é temporário, as operações são contínuas.

3) "O resultado de um projeto é exclusivo".

O protótipo de um produto é exclusivo. As unidades produzidas em escala não são exclusivas.
4) "Os projetos são únicos".

O conhecimento de conceitos, técnicas e ferramentas para o gerenciamento de projetos permitirá a condução de cada projeto de acordo com suas particularidades, porém, é necessário definir uma metodologia de base, seja oriunda dos guias de boas práticas ou própria. É possível que, para alguns tipos de projeto, uma versão reduzida das práticas descritas em determinado modelo mais algumas práticas específicas sejam suficientes. Criou-se uma "nova" metodologia.

A metodologia precisa ser seguida. Não significa prender o gerenciamento à metodologia, mas deve estar claro que a metodologia só deve ser alterada caso os benefícios para o projeto sejam claros ou fatores excepcionais obriguem. A adoção de uma metodologia e seu constante aprimoramento poderá conduzir ao padrão de excelência na gerência de projetos.

5) "Um projeto de sucesso é aquele que foi conduzido de acordo com o planejado e alcançou os resultados previstos".

Parece fácil, mas não é. Atingir os resultados é mais fácil. Realizar tudo como planejado requer conhecimento, experiência, metodologia e até sorte. Algumas vezes, fatores imprevistos impactam o projeto. Estes fatores podem ser positivos ou negativos para o projeto, até então, sorte. Mas, para aproveitar ou minimizar o efeito do impacto é preciso agir.

6) "O projeto não precisa exceder o previsto, caso aconteça, deve ser fruto da competência no gerenciamento ou da sorte, nunca do planejamento".

O planejamento perfeito obteria como resultado exatamente tudo o que foi planejado, considerando-se o desenvolvimento perfeito do projeto. Equívoco comum: considerar como fator de sucesso o projeto ter gasto apenas 75% do previsto. Foi um erro de planejamento que pode ter ocasionado diversos esforços desnecessários de planejamento, captação de recursos, estudos de viabilidade e mais, o projeto poderia ser descartado por ser considerado caro demais. Evidentemente, algo assim não pode ser considerado como fator de sucesso.

7) "Um stakeholder é uma pessoa, grupo ou instituição interessada no projeto".

É importante ressaltar que o interesse em um projeto pode ser muito diferente para cada stakeholder. Exemplos: o patrocinador deseja o retorno do investimento segundo suas pretensões, o gerente quer o sucesso do projeto, colaboradores querem recompensas e os opositores querem que o projeto fracasse. É fundamental conhecer todos os interessados no projeto e suas expectativas.

8) "A documentação deve conter todas as informações relevantes para o projeto e apenas isso".

Documentar é necessário. Alguns dos motivos: apresentar resultados aos stakeholders, facilitar o processo de mudanças, permitir maior integração e produtividade entre os envolvidos, atribuir responsabilidades, possibilitar o processo de melhoria contínua, etc.

Documentar sem necessidade pode desgastar a equipe, desperdiçar recursos, dificultar a utilização da própria documentação, etc.

Algumas perguntas básicas podem servir de guia. O documento:

- Agrega valor?
- Em quais situações será necessário?
- Sua importância justifica os recursos despendidos para documentar?

Como documentar é também importante. O foco deve ser a informação e sua utilização posterior. Por exemplo, é válido utilizar vários recursos para documentar com detalhes e boa formatação uma informação que será acessada constantemente ou de muita importância, porém para informações que serão acessadas raramente ou que são menos importantes a documentação deve consumir poucos recursos.

A bibliografia apresenta indicações dos principais guias ou modelos para o gerenciamento de projetos. Nesse livro, serão apresentadas as principais correntes de metodologias de gerenciamento de projetos: os métodos controlados e ágeis.

## 11.1 - EXERCÍCIO PROPOSTO

1) O que é um Projeto?

_____
_____
_____

## 12 - CONTROLE X AGILIDADE

"A gerência depende do controle".

As primeiras metodologias voltadas aos projetos também foram baseadas na metodologia científica, das quais herdaram sua principal característica: o controle. Essas metodologias baseiam-se em contratos, trabalham a partir de um planejamento completo, priorizam o controle e documentação, centralizam a responsabilidade no gerente do projeto e focam a entrega do produto final.

"Manifesto para Desenvolvimento Ágil de Software"

Indivíduos e interações entre eles mais que processos e ferramentas.

Software em funcionamento mais que documentação abrangente.

Colaboração com o cliente mais que negociação de contratos.

Responder a mudanças mais que seguir um plano.

Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda.

As metodologias ágeis surgiram posteriormente, especificamente entre desenvolvedores de Software. O principal motivo de seu surgimento foi a necessidade de adaptação dos projetos às mudanças. Necessitam da colaboração constante do cliente, utilizam planejamento por etapas, priorizam a execução, distribuem as responsabilidades e focam as entregas de cada etapa.

Apesar de alguns confrontos entre os adeptos de cada metodologia, o entendimento de ambas é desejável aos envolvidos em projetos. Cada qual tem seus pontos fortes, aplicações e limitações. Mas, alguns conceitos e ferramentas de ambas podem ser aplicadas aos projetos de desenvolvimento de Software, independentemente da linha mestra adotada.

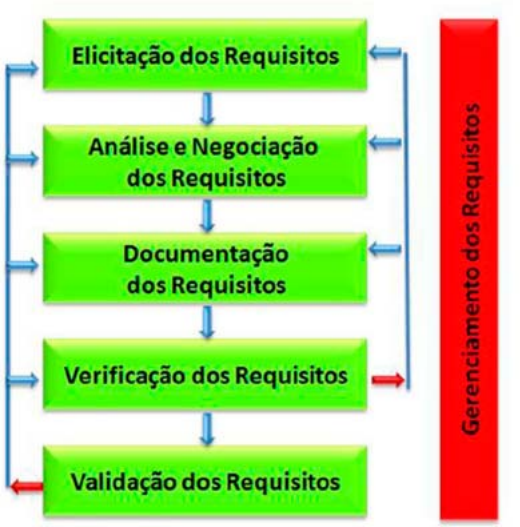
## 12.1 - EXERCÍCIO PROPOSTO

1) Qual a diferença entre Controlado e Ágil?

_____
_____
_____

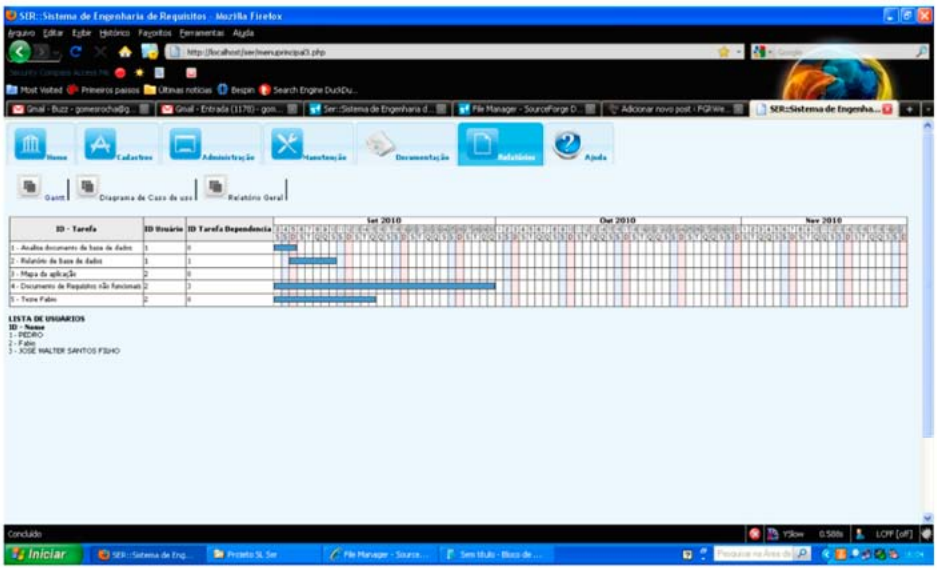
## 13 - SOFTWARE PARA ENGENHARIA DE REQUISITOS

Sabendo que a Engenharia de Requisitos é um dos principais pontos para o sucesso de um projeto, temos de fazer o máximo para que esta fase seja bem-sucedida. As fases que são partes da Engenharia de Requisitos são:



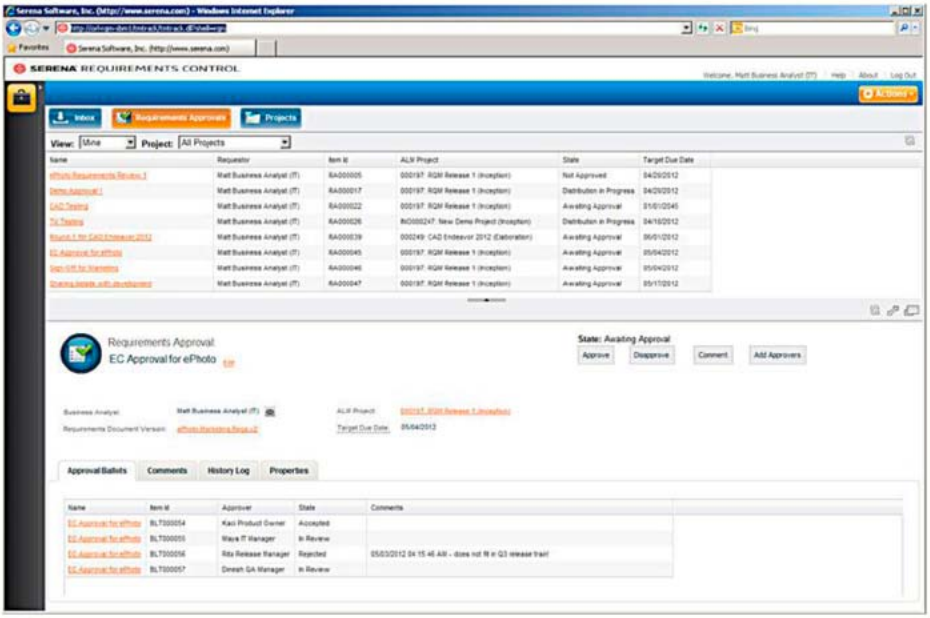

Desta forma, um Software para que possamos utilizar na fase de Engenharia de Requisitos, deve atender estas fases.

Além de poder gerenciar requisitos, este Software pode ser utilizado para gerenciar projetos controlados.



[1]Outro Software que pode ser utilizado para tal fim é o Serena, a empresa Serena é especializada em Softwares para Engenharia de Software e possui um Software que é gratuito disponível para Engenharia de Requisitos, ele está disponível em: http://www.serena.com/.

[1]O Projeto SER foi desenvolvido pelo professor Fábio Gomes Rocha.




Além destes Softwares, temos outros disponíveis, entre eles temos:

- IBM Rational RequisitePro
- Borland Caliber RM
- Requisite Manager

## 13.1 - EXERCÍCIO PROPOSTO

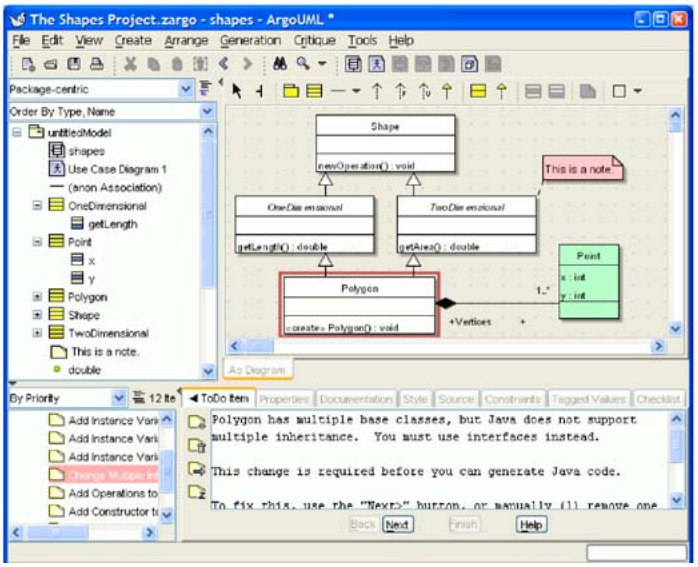
1) Quais as fases da Engenharia de Requisitos que são atendidos pelo projeto SER?

_____
_____
_____

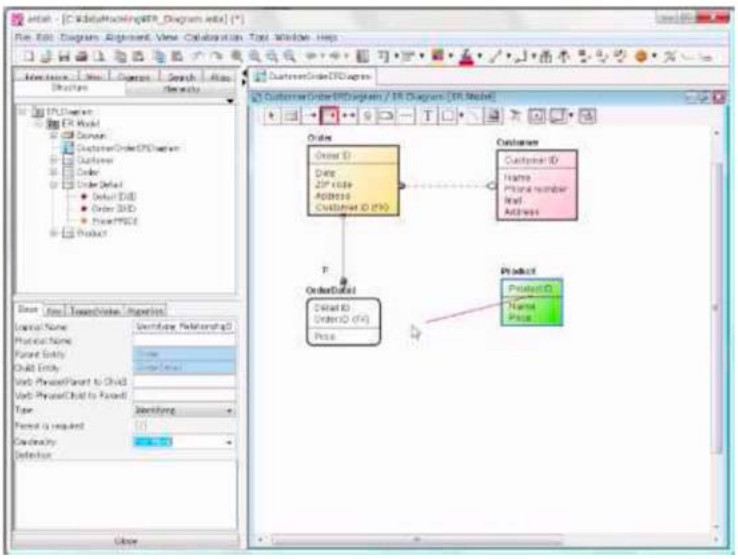
## 14 - SOFTWARE PARA UML

O conjunto de diagramas UML é importante para a Engenharia de Requisitos, principalmente no paradigma orientado a objetos. Desta forma, é importante a um bom analista e desenvolvedor conhecer as ferramentas para a criação de diagramas UML.

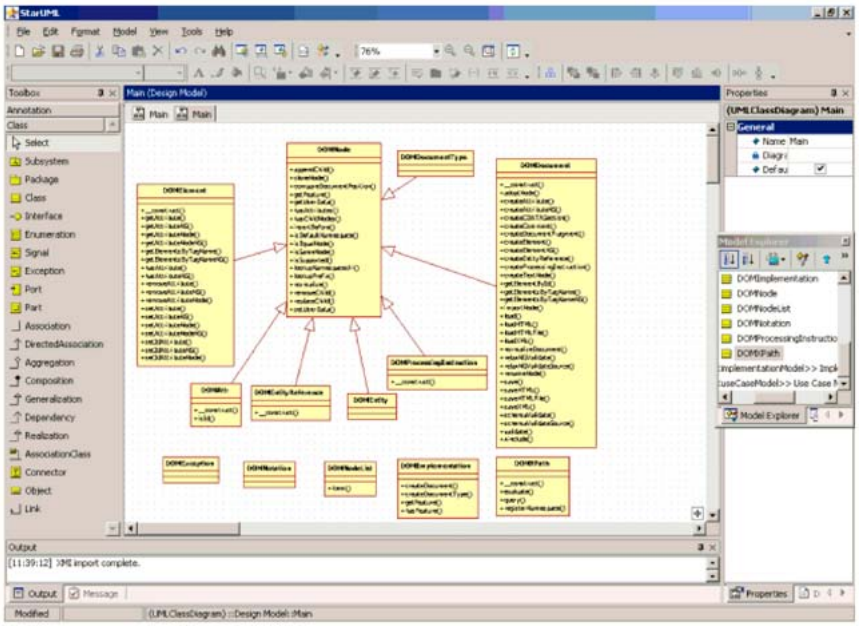
Existe uma infinidade de Softwares gratuitos e livres que possibilitam a diagramação UML, entre eles temos o ArgoUML, um Software gratuito, baseado no NetBeans para criação de diagramas UML. Ele está disponível no site: argouml.tigris.org/.


The Shapes Project.zargo - shapes - ArgoUML *
File Edit View Create Arrange Generation Critique Tools Help
Shape
This is a note.
Polygon
Point
Polygon has multiple base classes, but Java does not support multiple inheritance. You must use interfaces instead.
This change is required before you can generate Java code.
Next
Help

Além deste temos o Astah. Este não é bem um Software livre, é um Software gratuito, possuindo uma versão paga e uma versão comunitária que está disponível em astah.net/.



Além desta ferramenta temos o StarUML. Esta é uma ferramenta livre, multiplataforma que possibilita a criação de UML.



Além destas, existem ferramentas que suportam a criação de diagramas UML e ferramentas proprietárias para a criação de UML, sendo algumas:

- IBM Rational
- Borland Together
- Eclipse
- NetBeans

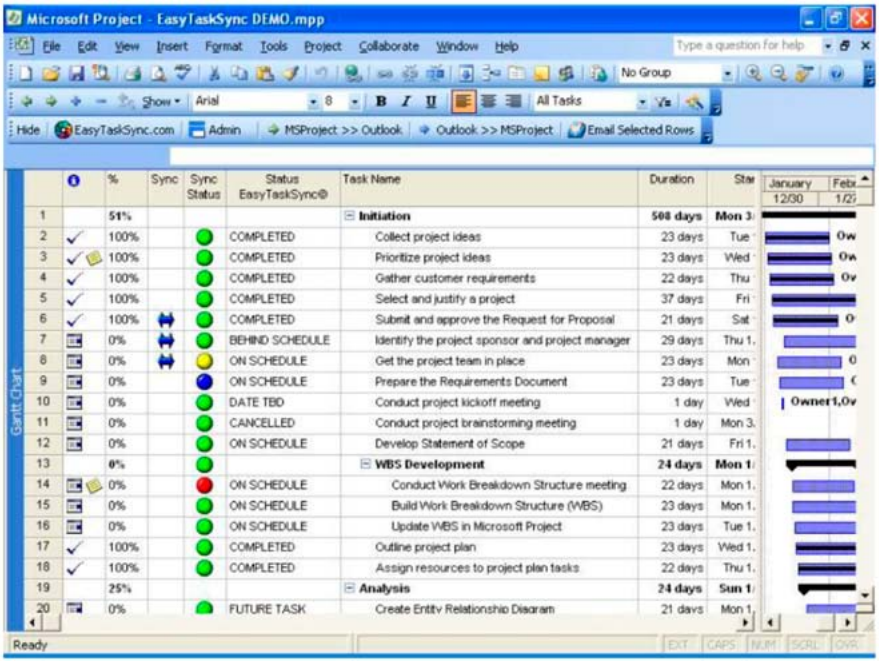
## 14.1 - EXERCÍCIO PROPOSTO

1) Faça uma pesquisa e cite pelo menos 4 ferramentas UML.

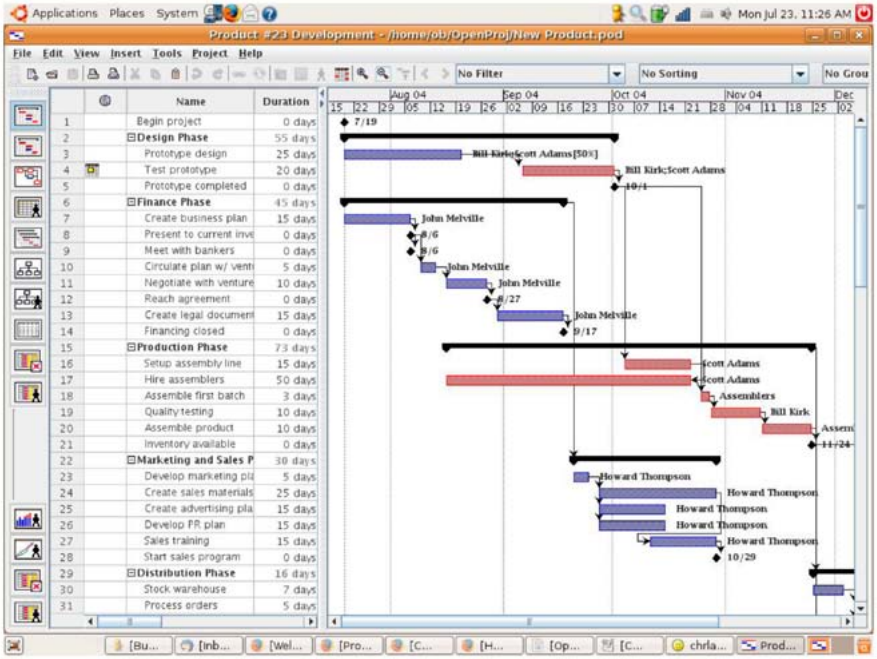
_____
_____
_____

## 15 - SOFTWARE PARA PROJETOS

Como vimos anteriormente, o projeto de Software incorpora diversas fases. É também um ponto crucial para o sucesso do Software e uma boa gestão, faz com que tenhamos sucesso no desenvolvimento, mas para isso é necessário termos informação e nada melhor do que um Software para a gestão dos projetos. Entre eles existem ferramentas proprietárias e ferramentas livres. A primeira ferramenta proprietária e bem conhecida é a Microsoft Project.

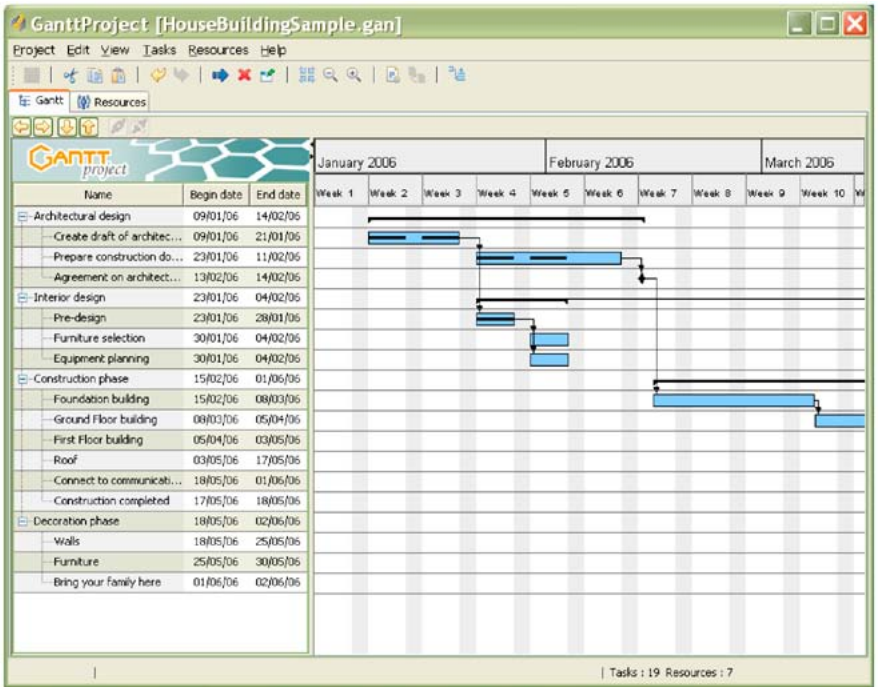

Microsoft Project - EasyTaskSync DEMO.mpp
File Edit View Insert Format Tools Project Collaborate Window Help
Gantt Chart
Ready

Esta é a ferramenta da Microsoft para gerenciamento de projetos, mas há uma infinidade de ferramentas livres, entre elas temos o OpenProj.

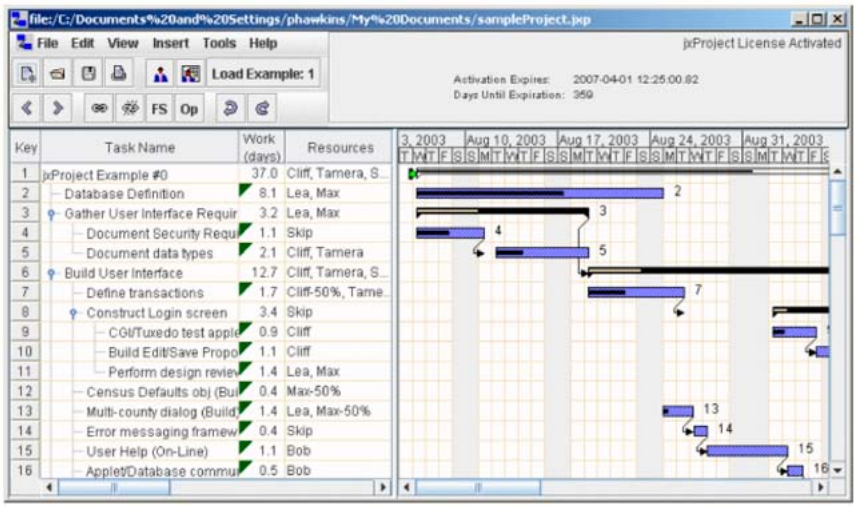

Applications Places System
Product #23 Development - /home/ob/OpenProj/New Product.pod
File Edit View Insert Tools Project Help

Ferramenta livre, multiplataforma e gratuita, o OpenProj é uma das ferramentas mais simples e práticas da atualidade.

Outro Software livre é o GanttProj, uma ferramenta simples e fácil de utilizar, multiplataforma.


GanttProject [HouseBuildingSample.gan]
Project Edit View Tasks Resources Help
Gantt Resources
January 2006
February 2006
March 2006
Name Begin date End date
Architectural design 09/01/06 14/02/06
Create draft of architec... 09/01/06 21/01/06
Prepare construction do... 23/01/06 11/02/06
Agreement on architect... 13/02/06 14/02/06
Interior design 23/01/06 04/02/06
Pre-design 23/01/06 28/01/06
Furniture selection 30/01/06 04/02/06
Equipment planning 30/01/06 04/02/06
Construction phase 15/02/06 01/06/06
Foundation building 15/02/06 08/03/06
Ground Floor building 08/03/06 05/04/06
First Floor building 05/04/06 03/05/06
Roof 03/05/06 17/05/06
Connect to communicati... 18/05/06 01/06/06
Construction completed 17/05/06 18/05/06
Decoration phase 18/05/06 02/06/06
Walls 18/05/06 25/05/06
Furniture 25/05/06 30/05/06
Bring your family here 01/06/06 02/06/06
Tasks : 19 Resources : 7

O jxProject é gratuito e feito em Java.


file:/C:/Documents%20and%20Settings/phawkins/My%20Documents/sampleProject.jxp
File Edit View Insert Tools Help
Load Example: 1
jxProject License Activated
Activation Expires: 2007-04-01 12:25:00.82
Days Until Expiration: 359
Key Task Name Work (days) Resources
1 jxProject Example #0 37.0 Cliff, Tamera, S...
2 Database Definition 8.1 Lea, Max
3 Gather User Interface Requir 3.2 Lea, Max
4 Document Security Requi 1.1 Skip
5 Document data types 2.1 Cliff, Tamera
6 Build User Interface 12.7 Cliff, Tamera, S...
7 Define transactions 1.7 Cliff-50%, Tame...
8 Construct Login screen 3.4 Skip
9 CGI/Tuxedo test appli 0.9 Cliff
10 Build Edit/Save Propo 1.1 Cliff
11 Perform design review 1.4 Lea, Max
12 Census Defaults obj (Bui 0.4 Max-50%
13 Multi-county dialog (Build 1.4 Lea, Max-50%
14 Error messaging framew 0.4 Skip
15 User Help (On-Line) 1.1 Bob
16 Applet/Database commu 0.5 Bob
3, 2003 Aug 10, 2003 Aug 17, 2003 Aug 24, 2003 Aug 31, 2003

Todas estas ferramentas possuem o gráfico de Gantt, que é um gráfico que facilita o entendimento das fases do projeto.

## 15.1 - EXERCÍCIO PROPOSTO

1) Faça uma pesquisa e cite pelo menos 4 ferramentas de Gestão de Projetos.

_____

_____

_____

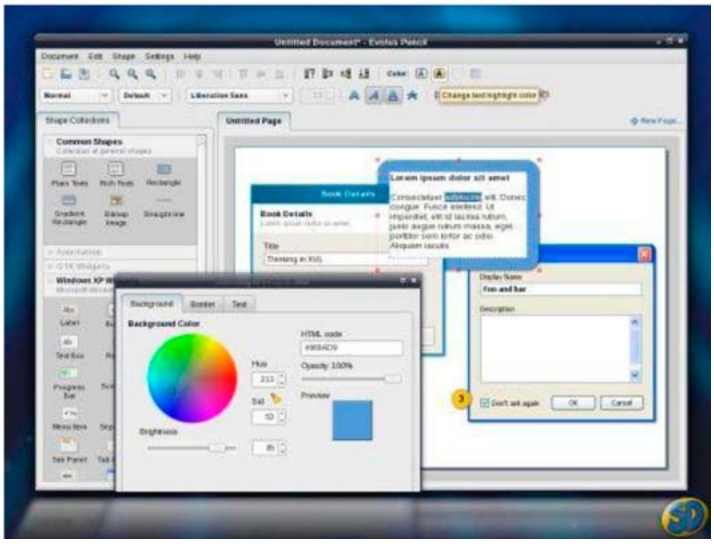
# Capítulo 16 - Software para Prototipação

## 16 - SOFTWARE PARA PROTOTIPAÇÃO

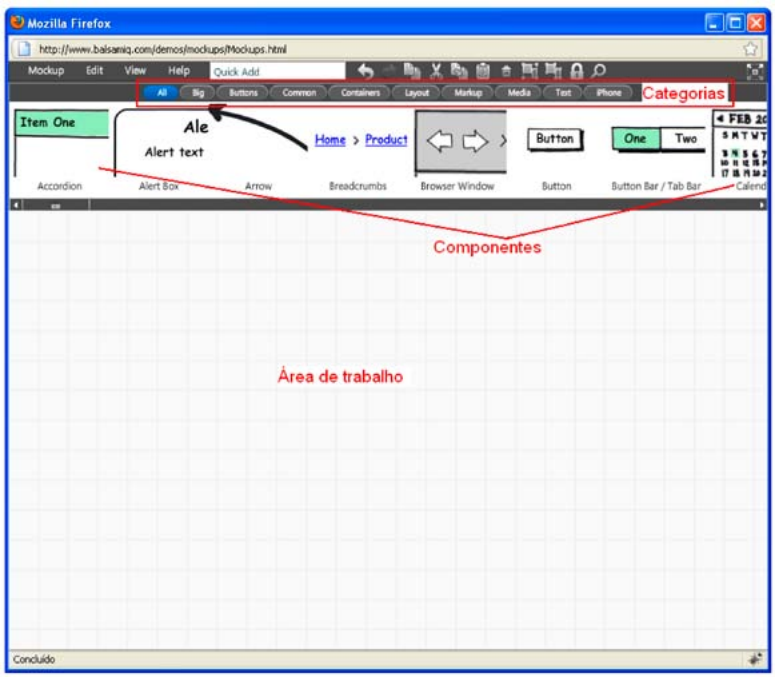
A prototipação é uma técnica muito eficiente para projetos de Software. Com base nisto, podemos desenvolver com maior agilidade e praticidade. Outro ponto importante é que a prototipação é uma técnica ágil e participativa, pois o cliente participa ativamente do desenvolvimento.

Existem Softwares para prototipação de tela, que na verdade apenas fazem o protótipo de tela. Entre elas temos:

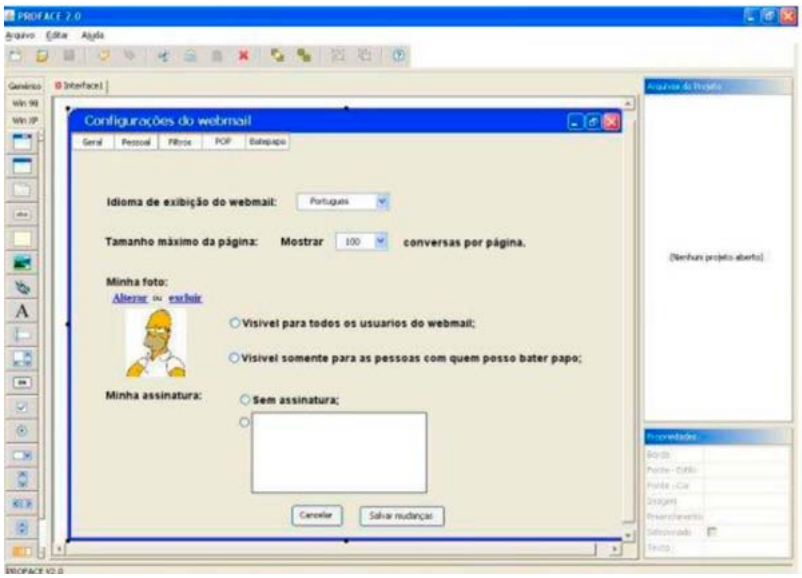
- **Pencil:** é um Software gratuito baseado no Firefox.



- **MxStudio**



- **ProFace:** é um software gratuito e livre.




É possível utilizar as ferramentas de desenvolvimento para criar os primeiros protótipos, como o Visual Studio, o Embarcadeiro Delphi e etc.

## 16.1 - EXERCÍCIO PROPOSTO

1) Faça uma pesquisa e cite pelo menos 2 Ferramentas de Prototipação?

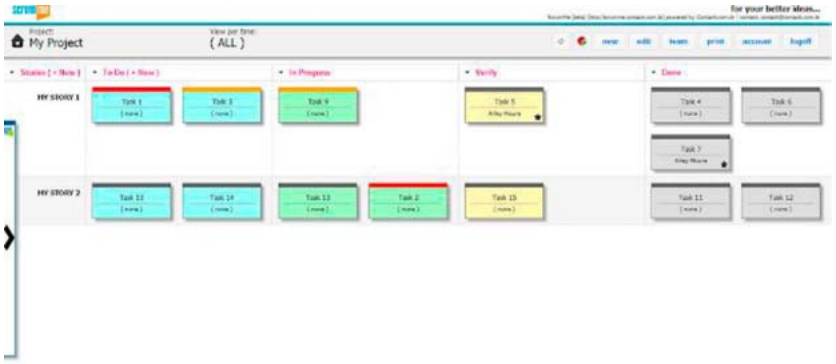
_____
_____
_____

## 17 - SOFTWARE PARA GESTÃO ÁGIL

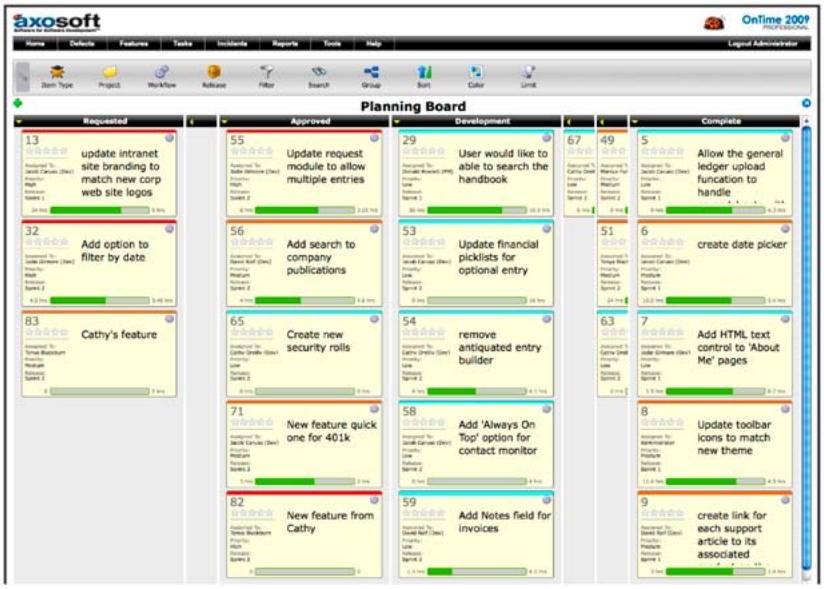
O gerenciamento ágil de projetos tem crescido constantemente, principalmente a metodologia Scrum, desta forma é importante conhecer ferramentas que possibilitem a gestão ágil de projetos.

Entre as ferramentas temos:

- **ScrumMe:** ferramenta gratuita e disponível em nuvem.




- **Scrum Planner**


axosoft
OnTime 2007
Planning Board
Requested
Approved
Development
Complete
update intranet site branding to match new corp web site logos
Add option to filter by date
Cathy's feature
Update request module to allow multiple entries
Add search to company publications
Create new security rolls
New feature quick one for 401k
New feature from Cathy
User would like to able to search the handbook
Update financial picklists for optional entry
remove antiquated entry builder
Add 'Always On Top' option for contact monitor
Add Notes field for invoices
Allow the general ledger upload funcation to handle
create date picker
Add HTML text control to 'About Me' pages
Update toolbar icons to match new theme
create link for each support article to its associated

- **ScrumHalf**

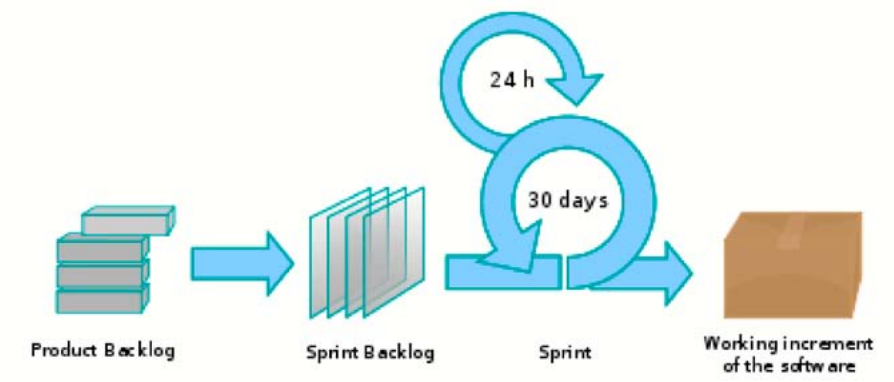
## 17.1 - EXERCÍCIO PROPOSTO

1) Faça uma pesquisa e cite pelo menos 2 Ferramentas de Kanban?

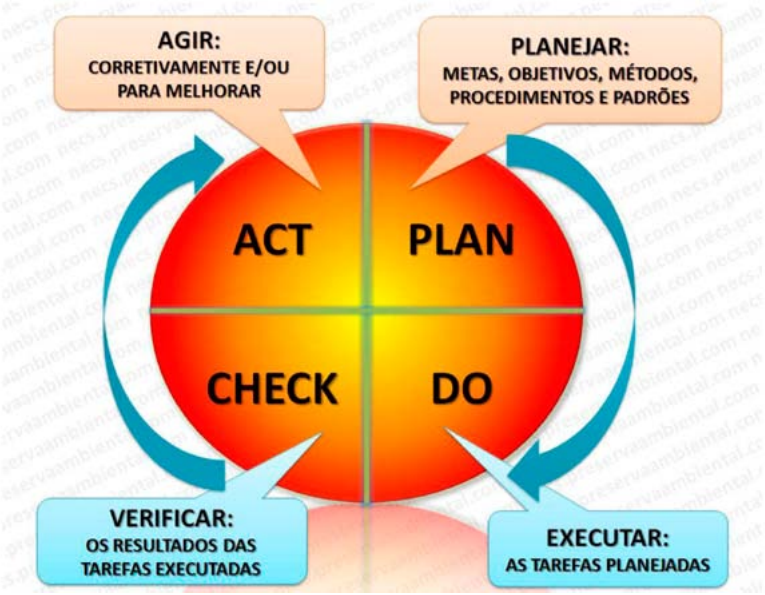
_____
_____
_____

## 18 - METODOLOGIA ÁGIL NA PRÁTICA

A metodologia ágil Scrum não é uma inovação, na verdade é a aplicação de boas práticas para o gerenciamento de projetos.



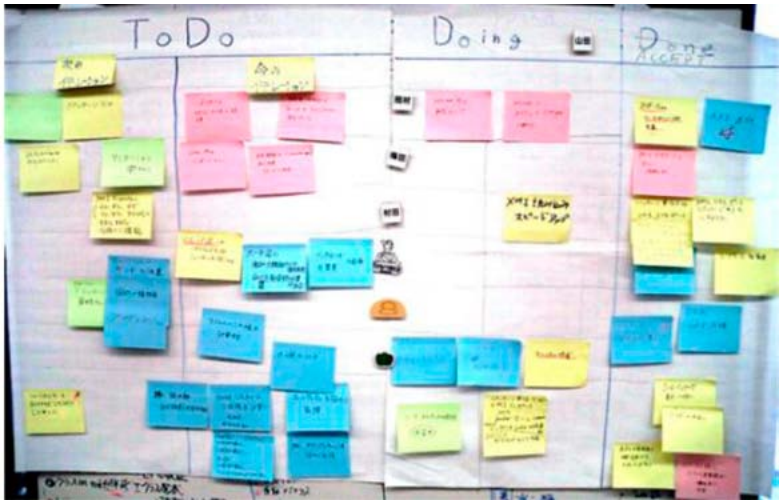

Para utilizar o Scrum, o principal ponto é a integração da equipe, pois não há divisão de papéis em times Scrum, ou seja, não temos um desenvolvedor, um testador, um analista, um DBA, todos fazem tudo em uma equipe Scrum. Outro ponto importante é que temos um ciclo PDCA constante, diariamente aplicado.




O ciclo PDCA diário evita problemas, pois diariamente é feito um planejamento e executadas as tarefas do dia.

No processo ágil temos o Sprint, que tem duração fixa. Cada Sprint deve possuir entregas, este é o ponto chave do Scrum, divide-se as tarefas de forma que seja possível desenvolver durante um Sprint.

A equipe é alto gerenciável e para isso é utilizado o Kanban, quadro em que as tarefas são fixadas e cada um da equipe pega a tarefa que achar mais apto a fazer, assim todos aprendem com o projeto.




O principal ponto no Scrum é a equipe de trabalho.

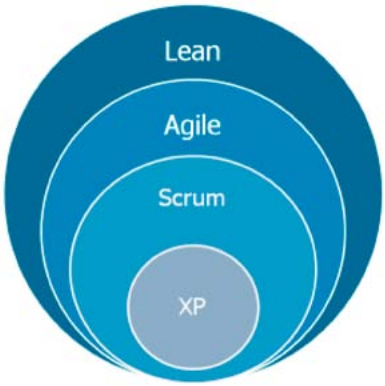
## 18.1 - EXERCÍCIO PROPOSTO

1) Faça uma pesquisa na Scrum Aliance sobre a Metodologia Scrum?

_____
_____
_____

## 19 - APLICANDO A METODOLOGIA ÁGIL

Para aplicar uma das metodologias ágeis temos que ter uma equipe coesa.

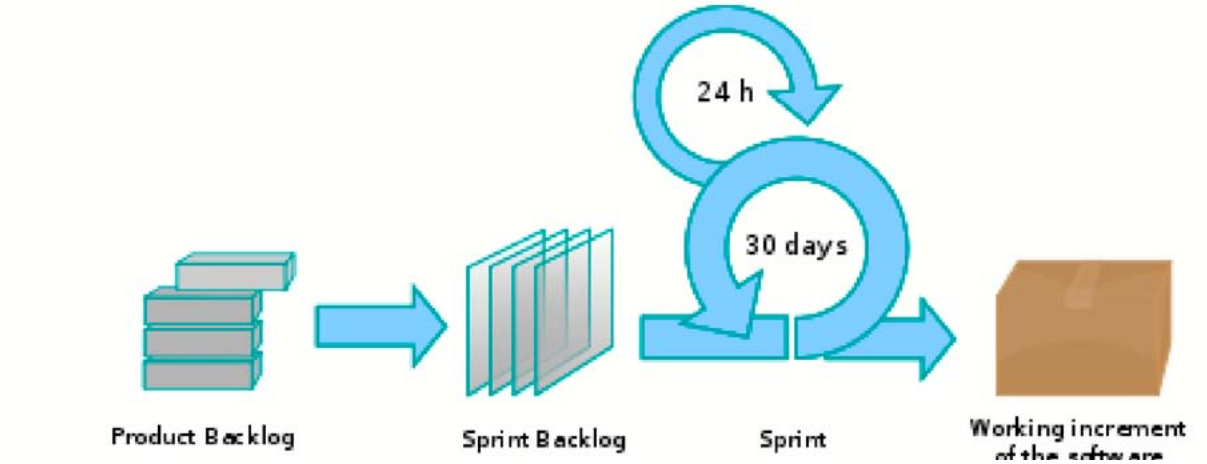



Todas as metodologias fixam em equipe, quanto melhor a equipe, mais ágil é o desenvolvimento.

Um dos pontos principais são as equipes pequenas. Evite equipes maiores que 13 participantes.

Para um perfeito funcionamento, temos que ter reuniões de início, reuniões diárias e reuniões finais, mas lembre-se que as reuniões são curtas.

Logo, temos reuniões de Sprint, reuniões diárias e reuniões finais.



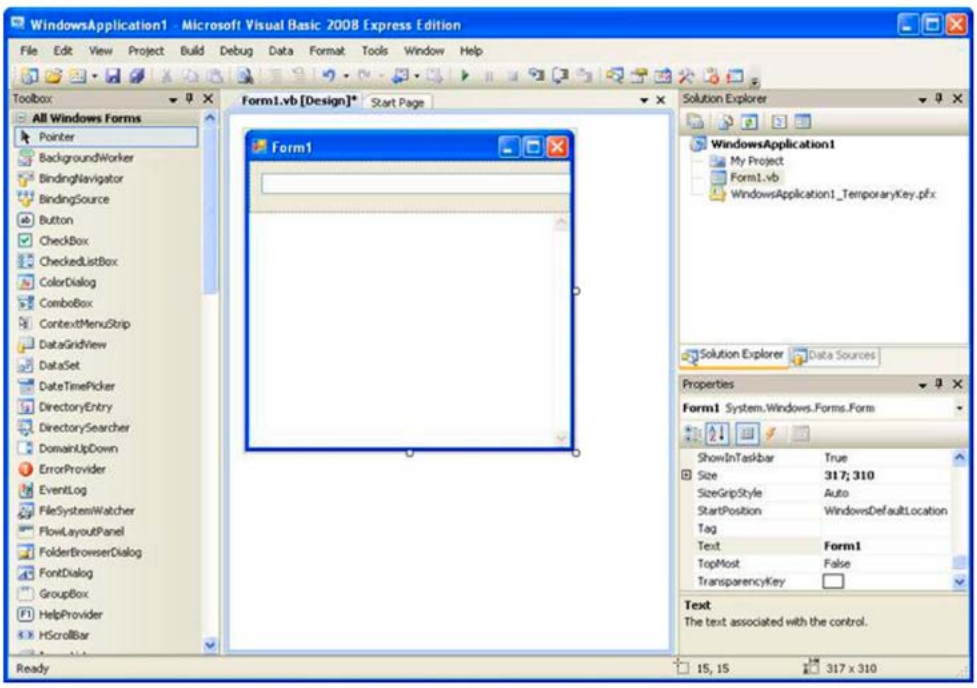
## 19.1 - EXERCÍCIO PROPOSTO

1) Faça uma pesquisa na Scrum Aliance sobre a Metodologia XP?

_____
_____
_____

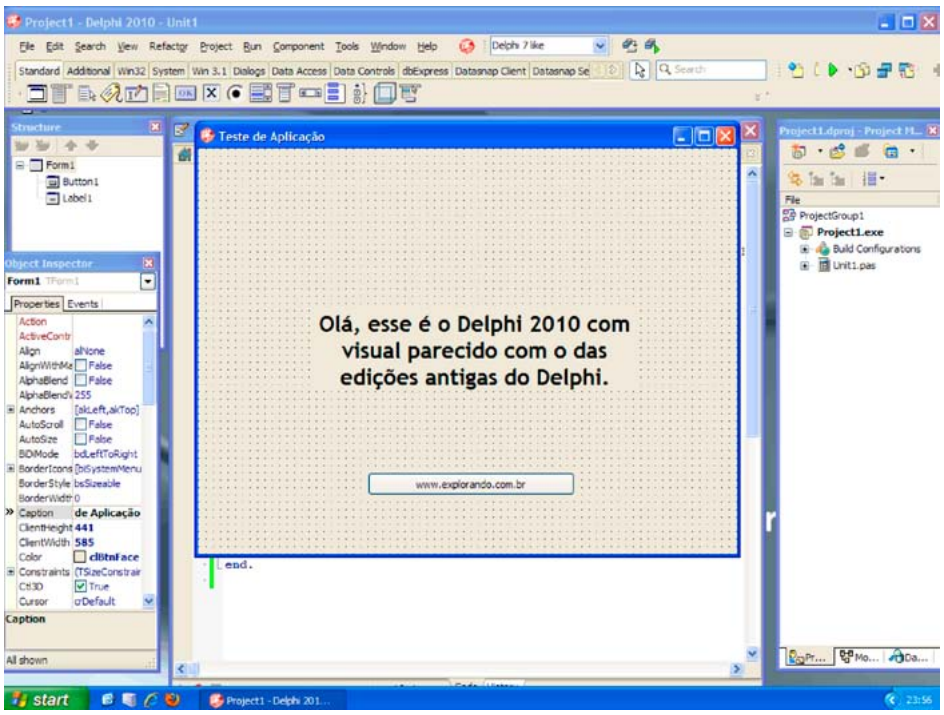
## 20 - FERRAMENTAS RAD

As ferramentas RAD são ferramentas de desenvolvimento rápido, elas têm o objetivo de facilitar o desenvolvimento de Software. Entre as ferramentas temos:

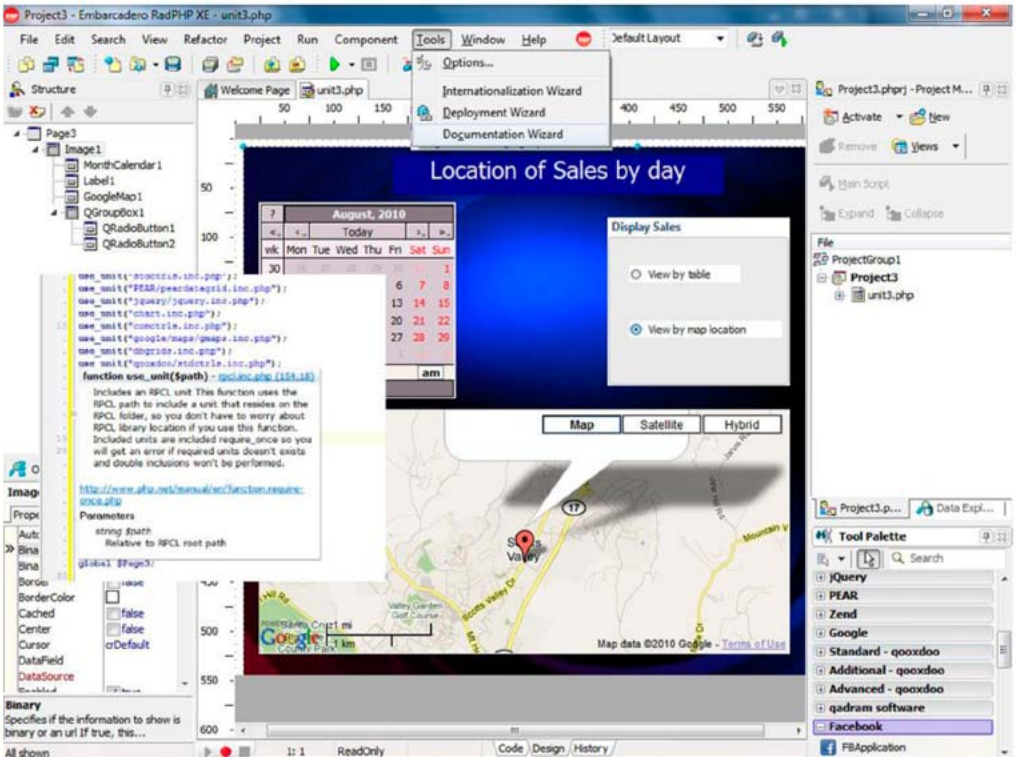



Basic de

• **Delphi:** é outra ferramenta de desenvolvimento ágil, mas para linguagem Objeto Pascal.



Já para desenvolvimento Web, temos o que era antigamente o Delphi for PHP, que agora se chama RadPHP.




Além desta ferramenta, temos:

- Eclipse
- NetBeans
- Dreamweaver

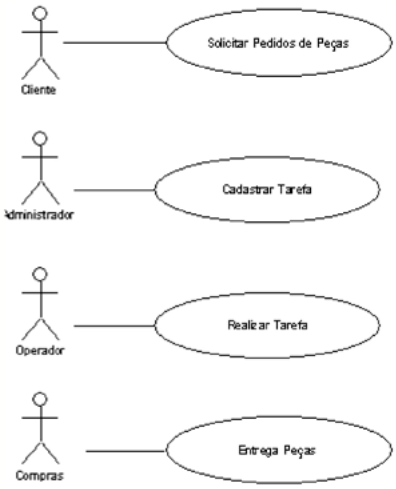
## 20.1 - EXERCÍCIO PROPOSTO

1) Faça uma pesquisa sobre outras ferramentas RAD?

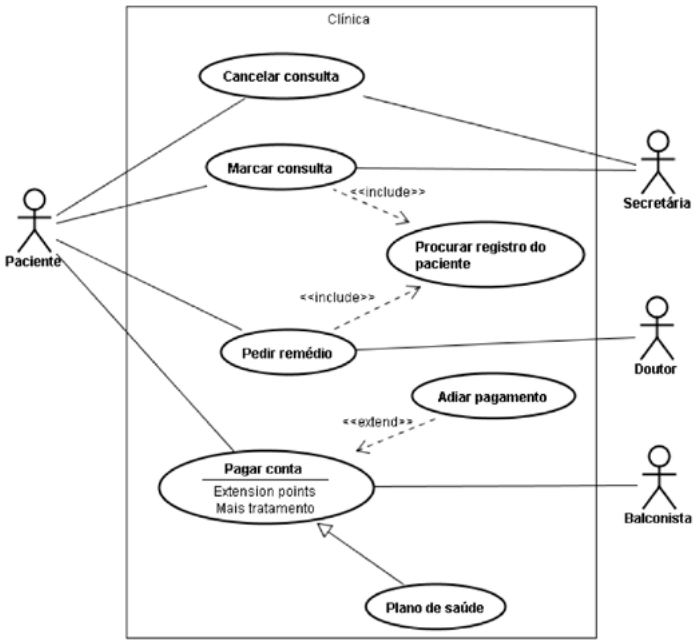
_____
_____
_____

## 21 - UTILIZANDO CASO DE USO

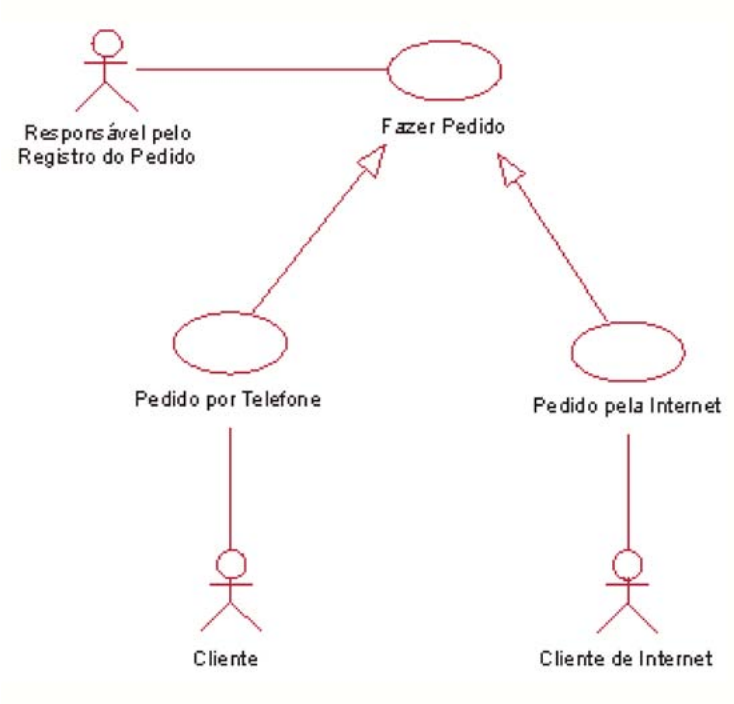
Dentro da UML existem diversos diagramas e entre eles temos o Caso de Uso, que é um dos principais diagramas para o levantamento de caso de uso. O caso de uso é composto por atores, ligações e tarefas. Os atores são representados por um boneco e a tarefa é representada por um eclipse.




No exemplo anterior temos diversos atores e diversas tarefas.




As tarefas podem ser ligadas diretamente ou indiretamente como no exemplo anterior.


Responsável pelo
Registro do Pedido
Fazer Pedido
Pedido por Telefone
Pedido pela Internet
Cliente
Cliente de Internet

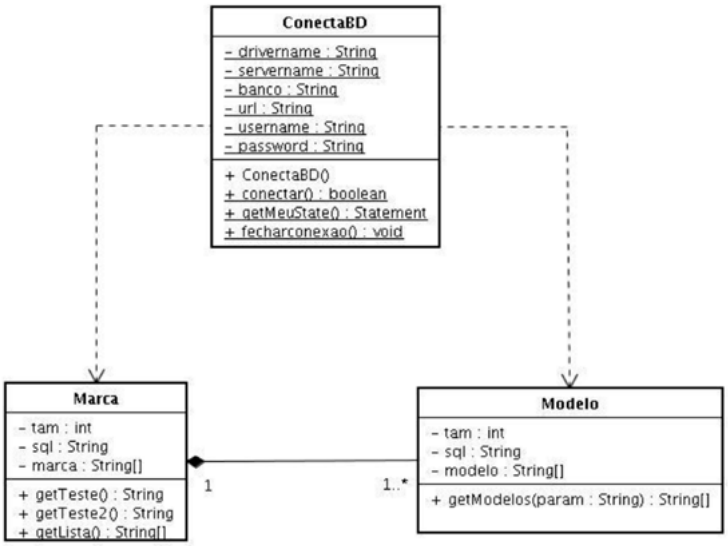
## 21.1 - EXERCÍCIO PROPOSTO

1) Faça uma pesquisa sobre os Diagramas UML – Caso de Uso?

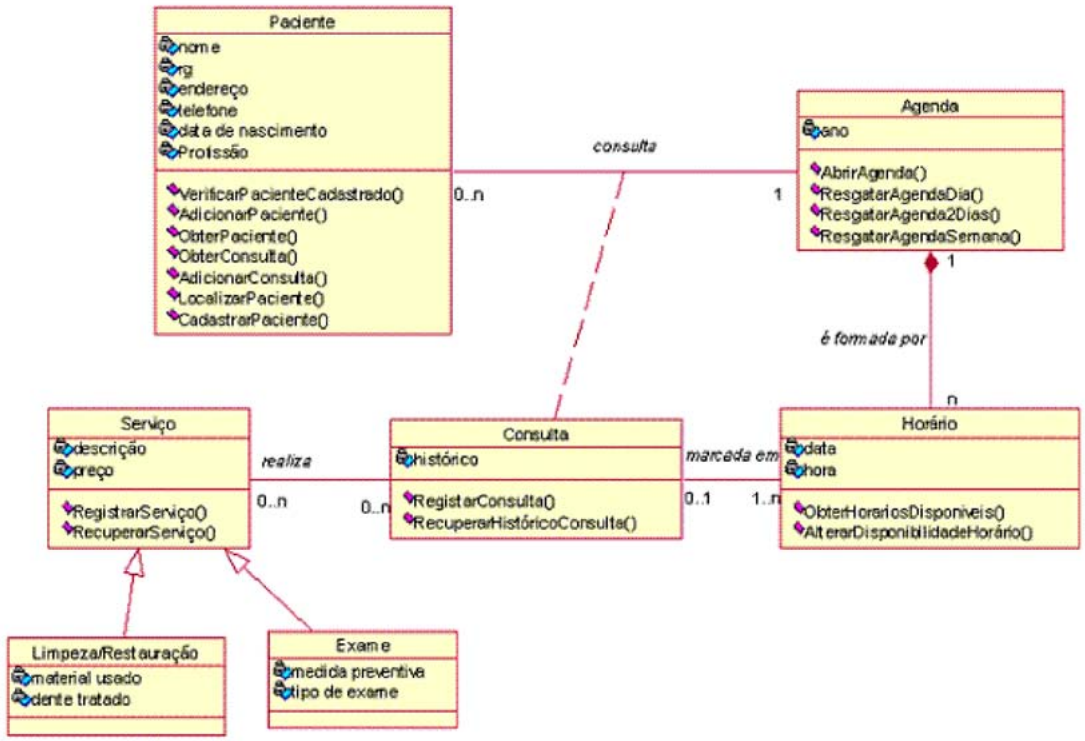
_____
_____
_____

## 22 - UTILIZANDO UML – CLASSE

Dentro da UML, o Diagrama de Classe é provavelmente um dos mais importantes diagramas, pois é utilizado para a orientação a objetos.

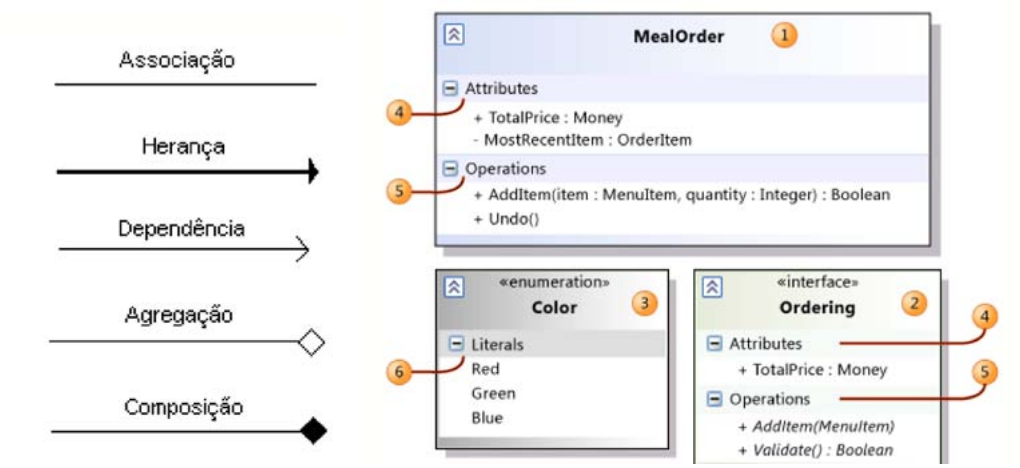


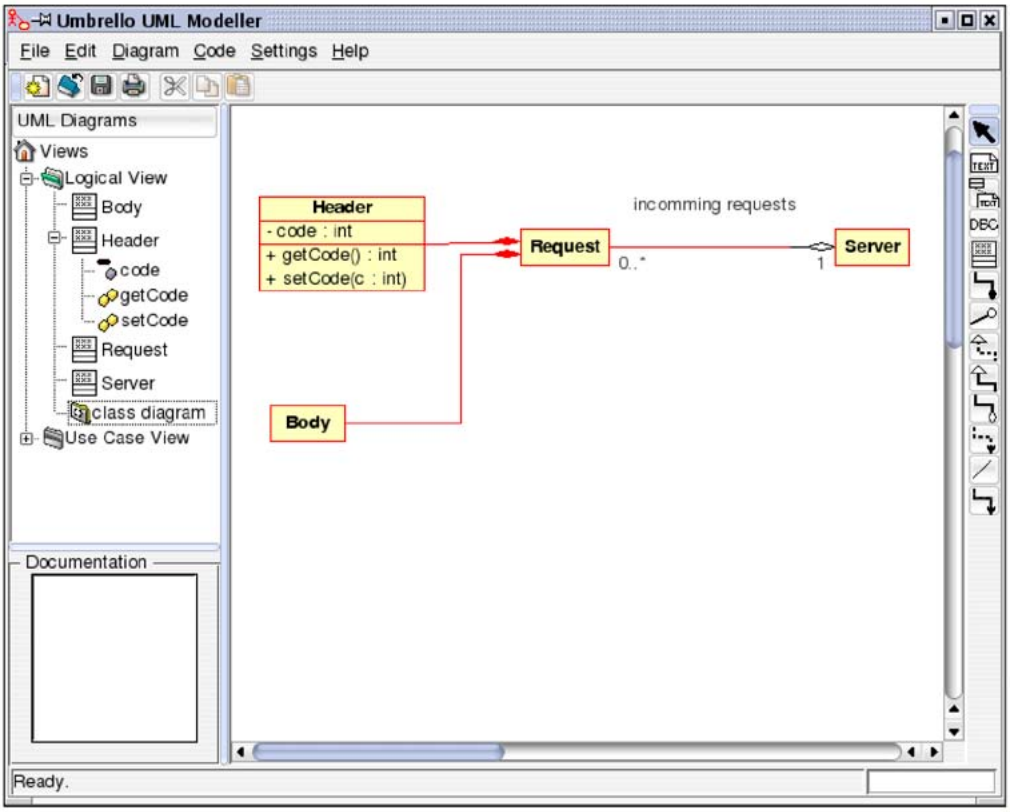

Note no exemplo acima que temos três classes. As classes possuem atributos e métodos separados por linhas, facilitando o entendimento.




Aqui temos um exemplo mais completo.

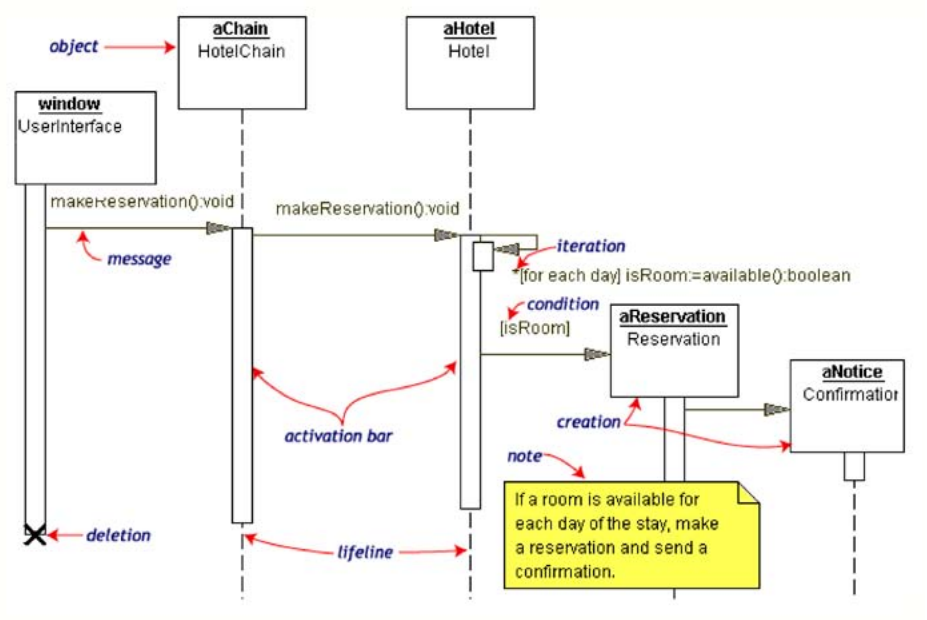
As ligações podem ser de herança, dependência, agregação e etc.


Associação
Herança
Dependência
Agregação
Composição
MealOrder
1
Attributes
+ TotalPrice : Money
- MostRecentItem : OrderItem
Operations
+ AddItem(item : MenuItem, quantity : Integer) : Boolean
+ Undo()
«enumeration»
Color
3
Literals
Red
Green
Blue
«interface»
Ordering
2
Attributes
+ TotalPrice : Money
Operations
+ AddItem(MenuItem)
+ Validate() : Boolean
4
5
6

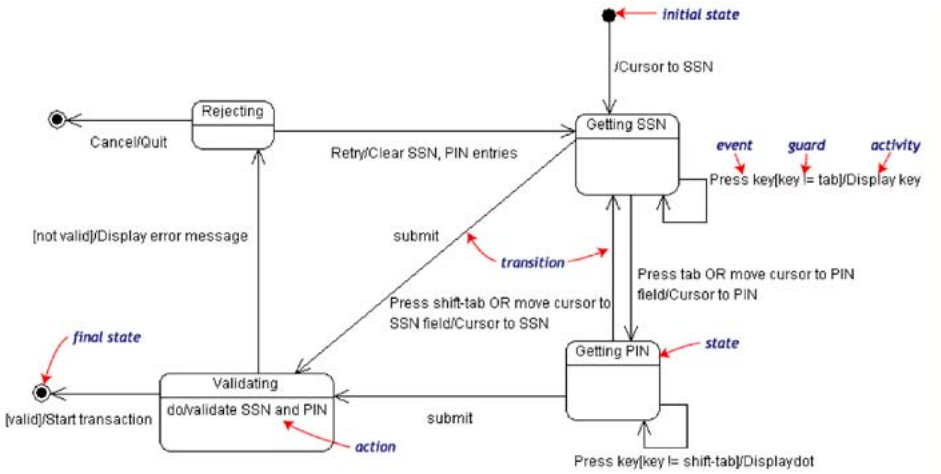

Umbrello UML Modeller
File Edit Diagram Code Settings Help
UML Diagrams
Views
Logical View
Body
Header
code
getCode
setCode
Request
Server
class diagram
Use Case View
Documentation
Header
- code : int
+ getCode() : int
+ setCode(c : int)
incomming requests
Request
0..*
1
Server
Body
Ready.

## 23 - UTILIZANDO UML – DIAGRAMAS

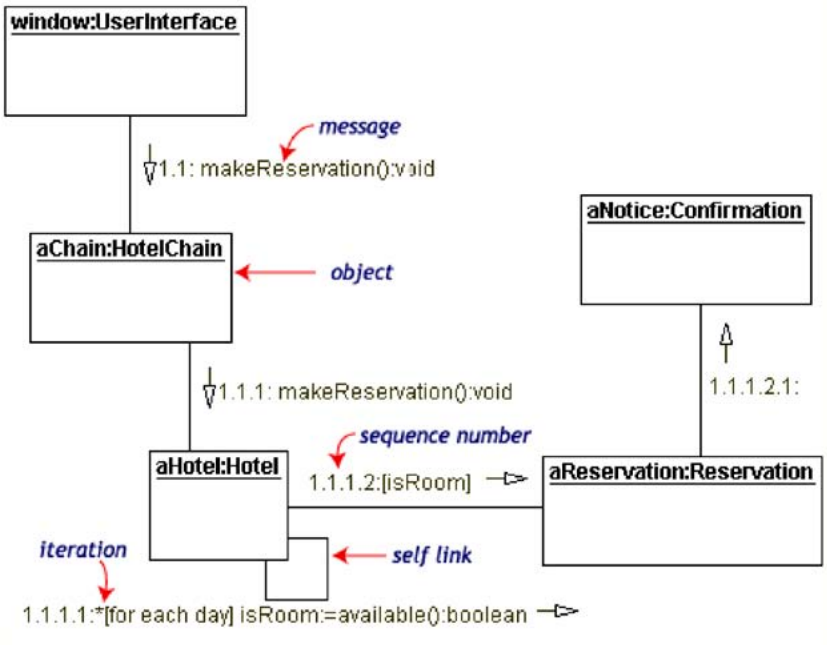
Além dos diagramas de Caso de Uso e de Classes, a UML disponibiliza diversos diagramas, entre eles temos o Diagrama de Sequência, que tem por objetivo mostrar a sequência da comunicação entre os objetos.



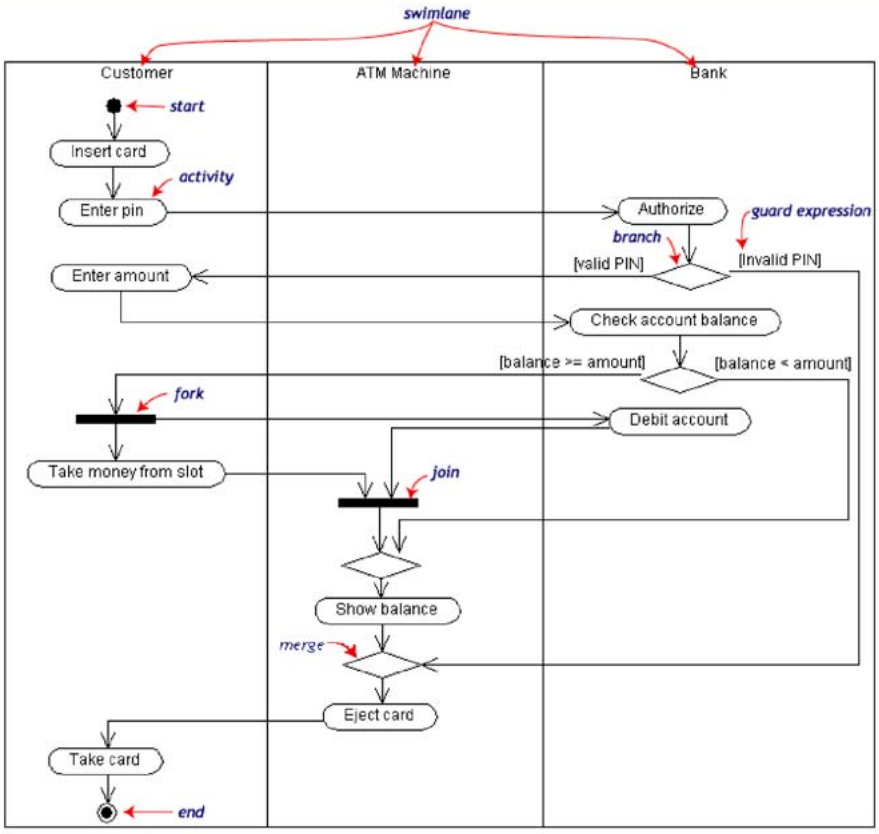

O Diagrama de Estado demonstra o estado do objeto.




O Diagrama de Interação apresenta uma forma simples de interação entre componentes de Software.




E por fim, o Diagrama de Atividade, que completa o Diagrama de Sequência, demonstrando como as atividades vão funcionar e se comunicar.



## 23.1 - EXERCÍCIO PROPOSTO

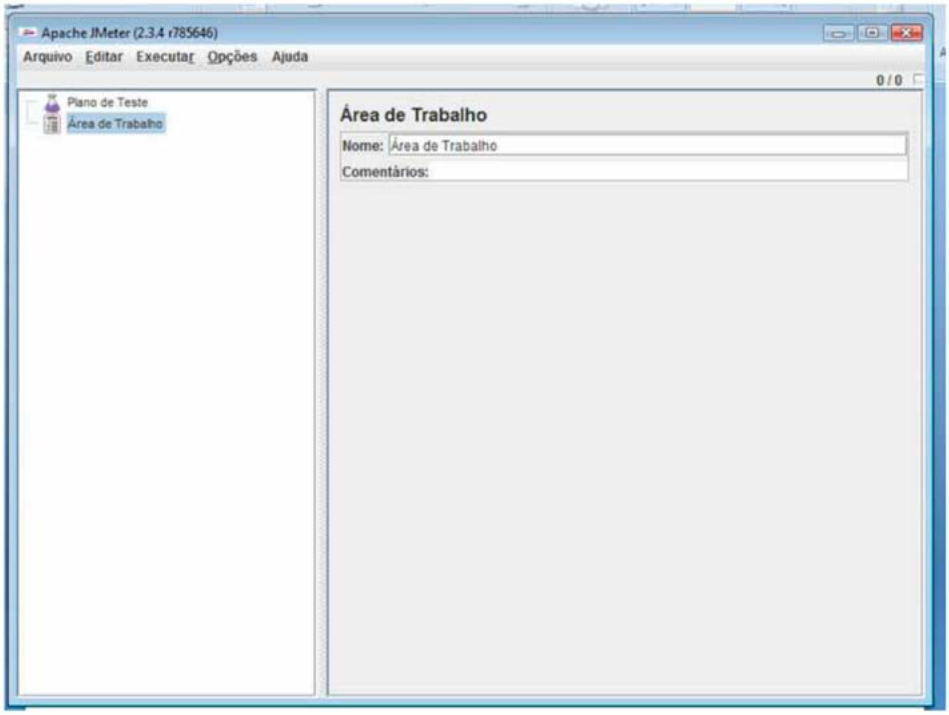
1) Faça uma pesquisa sobre os Diagramas UML?

_____
_____
_____

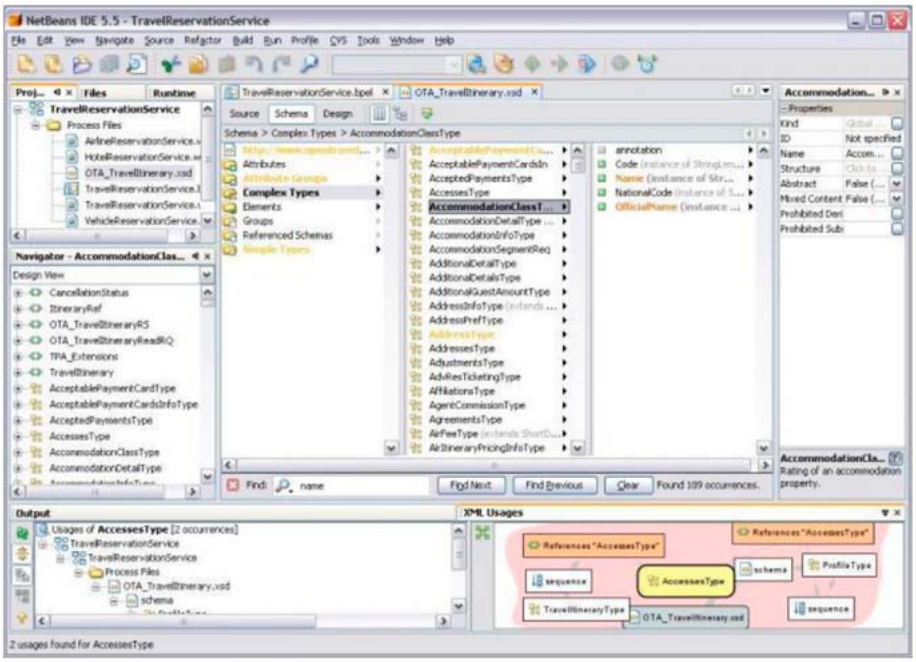
## 24 - OUTRAS FERRAMENTAS

Existem diversas ferramentas para desenvolvimento, engenharia e teste.

O JMeter é uma ferramenta livre para teste de performance de sistemas, está disponível gratuitamente, foi criada em Java e é multiplataforma.

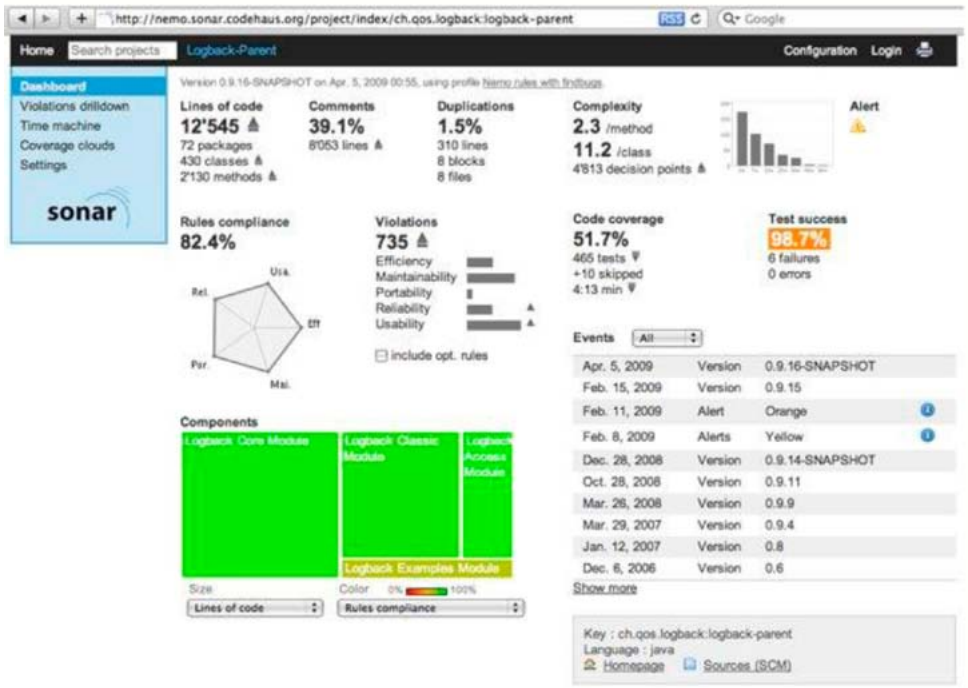



NetBeans é outra ferramenta para desenvolvimento e pode ser utilizada para desenvolver Java, PHP e etc.

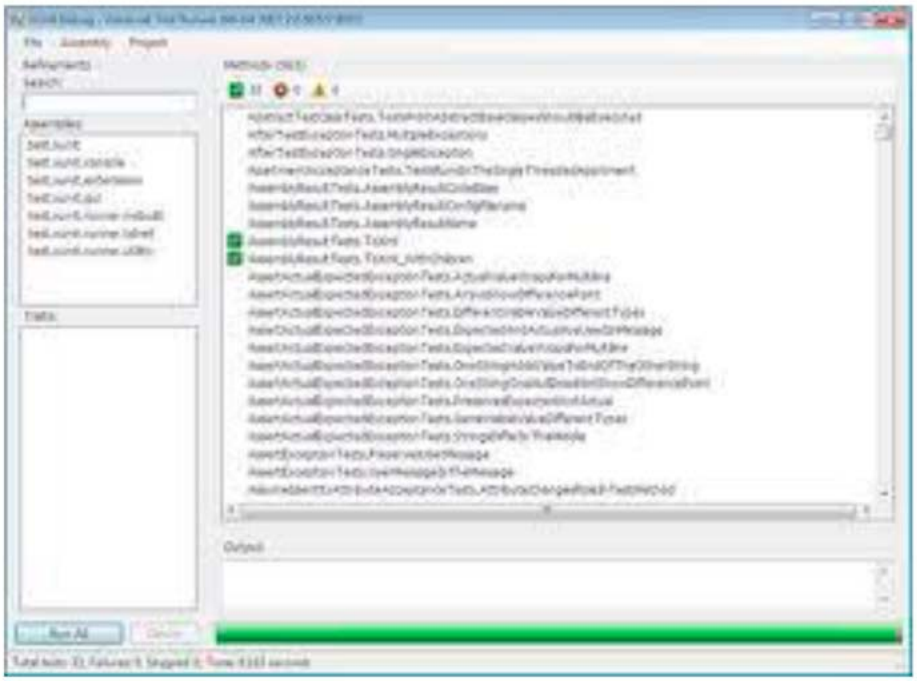



- Esta ferramenta é gratuita e livre e está disponível para download.

Existem ainda ferramentas de integração contínua, que são muito utilizadas em metodologias ágeis.




E não podendo faltar as ferramentas de testes unitários ou xUnit´s.



Existe uma infinidade de ferramentas de teste unitário, entre elas temos o PHPUnit, JUnit e etc.

# Respostas dos exercícios

## Capítulo 1

**1) Qual o principal problema em Projetos de Software?**

R: A falta de planejamento.

## Capítulo 2

**1) O que é Engenharia de Software?**

R: A Engenharia de Software (ES) é um ramo da computação que tem como principal objetivo solucionar os problemas práticos do mundo real através da aplicação de conhecimentos científicos.

## Capítulo 3

**1) Quais são as fases do Ciclo de Vida Cascata?**

R: Este modelo divide-se nas fases: Levantamento de Requisitos e Análise, Projeto de Sistema, Implementação, Testes, Implantação de Sistema e Manutenção.

## Capítulo 4

**1) Qual a principal vantagem da Prototipação?**

R: A principal vantagem da prototipação é reduzir as falhas de requisitos.

## Capítulo 6

**1) Quais são as leis que regem o Sistemismo?**

R: Todo sistema é composto de subsistemas.

- Todo sistema é parte de um sistema maior.
- Quanto maior a divisão de subsistemas, maior a necessidade de coordenação de partes.
- Homeostase.
- Sinergia.

## Capítulo 7

**1) Quais as etapas da Engenharia de Requisitos?**

R: Elicitação de requisitos

- Análise de requisitos
- Especificação de requisitos
- Validação de requisitos
- Gestão de requisitos

## Capítulo 8

**1) O que é Elicitação de Requisitos?**

R: A Elicitação de Requisitos é a descoberta das necessidades do produto.

## Capítulo 9

**1) Quais são as Técnicas para Elicitação de Requisitos?**

R: Entrevista, Brainstorming e Uso de UML.
Capítulo 10

**2) Quais são as Técnicas de Análise mais conhecidas?**

R: Técnicas de Análise Estruturada e Técnica Orientada a Objetos.
Capítulo 11

**3) O que é um Projeto?**

R: Um projeto é um conjunto temporário de esforços objetivando um resultado exclusivo.

## Capítulo 12

**1) Qual a diferença entre Controlado e Ágil?**

R: O controlado baseia-se em contratos, trabalha a partir de um planejamento completo, prioriza o controle e documentação, centraliza a responsabilidade no gerente do projeto e foca a entrega do produto final. Já o ágil necessita da colaboração constante do cliente, utiliza planejamento por etapas, prioriza a execução, distribui as responsabilidades e foca as entregas de cada etapa.

## Capítulo 13

**1) Quais as fases da Engenharia de Requisitos que são atendidos pelo projeto SER?**

R: Elicitação de Requisitos, Análise e Negociação dos Requisitos, Documentação dos Requisitos, Verificação dos Requisitos e Validação dos Requisitos.

## Capítulo 14

**1) Faça uma pesquisa e cite pelo menos 4 ferramentas UML.**

R: Resposta Pessoal.

## Capítulo 15

**1) Faça uma pesquisa e cite pelo menos 4 ferramentas de Gestão de Projetos.**

R: Resposta Pessoal.

## Capítulo 16

**1) Faça uma pesquisa e cite pelo menos 2 Ferramentas de Prototipação?**

R: Resposta Pessoal.

## Capítulo 17

**1) Faça uma pesquisa e cite pelo menos 2 Ferramentas de Kanban?**

R: Resposta Pessoal.

## Capítulo 18

**1) Faça uma pesquisa na Scrum Aliance sobre a Metodologia Scrum?**

R: Resposta Pessoal.

## Capítulo 19

**1) Faça uma pesquisa na Scrum Aliance sobre a Metodologia XP?**

R: Resposta Pessoal.

## Capítulo 20

**1) Faça uma pesquisa sobre outras ferramentas RAD?**

R: Resposta Pessoal.

## Capítulo 21

**1) Faça uma pesquisa sobre os Diagramas UML – Caso de Uso?**

R: Resposta Pessoal.

## Capítulo 23

**1) Faça uma pesquisa sobre os Diagramas UML?**

R: Resposta Pessoal.

# Referências


BARTIÉ, Alexandre. *Garantia de Qualidade de Software*. Rio de Janeiro: Campus, 2002.
Ciclo de Vida Cascata. Disponível em: http://www.dgz.org.br/jun09/Art_03.htm. Acessado em: 05/06/2012.

Ciclo de Vida Espiral. Disponível em: http://www2.dem.inpe.br/ijar/CicoloVidaSoftPrado.html. Acessado em: 05/06/2012.

COHN, Mike. *Desenvolvimento de Software com Scrum: aplicando métodos ágeis com sucesso.* Porto Alegre: Bookman, 2011.

Manifesto para Desenvolvimento *Ágil de Software.* Disponível em: http://agilemanifesto.org/iso/ptbr/. Acessado em: 05/06/2012.

MARTINS, José Carlos Cordeiro. *Gestão de Projetos de Desenvolvimento de Software*. Rio de Janeiro: Brasport, 2001.

Processo de Engenharia de Requisitos. Disponível em: http://www.devmedia.com.br/space.asp?id=216958. Acessado em: 05/06/2012.

Processo de Produção de Software. Disponível em: http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/intro/processo.htm. Acessado em: Acessado em: 05/06/2012.

Processos de Engenharia de Software. Disponível em: http://www.pedrofcarvalho.com.br/engenharia.html. Acessado em: 05/06/2012.

Prototipação. Disponível em: http://nuxabril.wordpress.com/2009/06/24/nucleo-de-user-experience/. Acessado em: 05/06/2012.

Sourceforge. Disponível em: http://sourceforge.net/p/sersistemadeeng/home/Home/. Acessado em: 05/06/2012.

alfamacursos.com.br